

Operating Systems

Lecture 14: Advanced Paging + Swapping

Nipun Batra

Sep 4, 2018

Revision - Picture Thus Far

Revision - Picture Thus Far

Revision - Picture Thus Far

- Base & Bounds

Revision - Picture Thus Far

- Base & Bounds
 - Pros: Very quick, 2 registers

Revision - Picture Thus Far

- Base & Bounds
 - Pros: Very quick, 2 registers
 - Cons: Contiguous block of memory -> fragmentation

Revision - Picture Thus Far

- Base & Bounds
 - Pros: Very quick, 2 registers
 - Cons: Contiguous block of memory -> fragmentation
- Segmentation

Revision - Picture Thus Far

- Base & Bounds
 - Pros: Very quick, 2 registers
 - Cons: Contiguous block of memory -> fragmentation
- Segmentation
 - Pros: Still relatively simple, 3 registers, lesser fragmentation

Revision - Picture Thus Far

- Base & Bounds
 - Pros: Very quick, 2 registers
 - Cons: Contiguous block of memory -> fragmentation
- Segmentation
 - Pros: Still relatively simple, 3 registers, lesser fragmentation
 - Cons: Still contiguous block of memory for segment

Revision - Picture Thus Far

- Base & Bounds
 - Pros: Very quick, 2 registers
 - Cons: Contiguous block of memory -> fragmentation
- Segmentation
 - Pros: Still relatively simple, 3 registers, lesser fragmentation
 - Cons: Still contiguous block of memory for segment
- Paging

Revision - Picture Thus Far

- Base & Bounds
 - Pros: Very quick, 2 registers
 - Cons: Contiguous block of memory -> fragmentation
- Segmentation
 - Pros: Still relatively simple, 3 registers, lesser fragmentation
 - Cons: Still contiguous block of memory for segment
- Paging
 - Pros: Very low chances of segmentation

Revision - Picture Thus Far

- Base & Bounds
 - Pros: Very quick, 2 registers
 - Cons: Contiguous block of memory -> fragmentation
- Segmentation
 - Pros: Still relatively simple, 3 registers, lesser fragmentation
 - Cons: Still contiguous block of memory for segment
- Paging
 - Pros: Very low chances of segmentation
 - Cons: Slow, lots of memory accesses; memory overhead/
process is huge!

Revision - Picture Thus Far

- Base & Bounds
 - Pros: Very quick, 2 registers
 - Cons: Contiguous block of memory -> fragmentation
- Segmentation
 - Pros: Still relatively simple, 3 registers, lesser fragmentation
 - Cons: Still contiguous block of memory for segment
- Paging
 - Pros: Very low chances of segmentation
 - Cons: Slow, lots of memory accesses; memory overhead/
process is huge!
- Paging + TLB

Revision - Picture Thus Far

- Base & Bounds
 - Pros: Very quick, 2 registers
 - Cons: Contiguous block of memory -> fragmentation
- Segmentation
 - Pros: Still relatively simple, 3 registers, lesser fragmentation
 - Cons: Still contiguous block of memory for segment
- Paging
 - Pros: Very low chances of segmentation
 - Cons: Slow, lots of memory accesses; memory overhead/ process is huge!
- Paging + TLB
 - Pros: Improves the address translation speed (spatial & temporal locality)

Revision - Picture Thus Far

- Base & Bounds
 - Pros: Very quick, 2 registers
 - Cons: Contiguous block of memory -> fragmentation
- Segmentation
 - Pros: Still relatively simple, 3 registers, lesser fragmentation
 - Cons: Still contiguous block of memory for segment
- Paging
 - Pros: Very low chances of segmentation
 - Cons: Slow, lots of memory accesses; memory overhead/process is huge!
- Paging + TLB
 - Pros: Improves the address translation speed (spatial & temporal locality)
 - Cons: Limited in size, memory overhead/process still huge

Revision - Reducing Memory Overheads of Paging

Revision - Reducing Memory Overheads of Paging

- 32 bit address space with 4 KB pages

Revision - Reducing Memory Overheads of Paging

- 32 bit address space with 4 KB pages
- 4 KB pages -> 12 bits?

Revision - Reducing Memory Overheads of Paging

- 32 bit address space with 4 KB pages
- 4 KB pages \rightarrow 12 bits?
- Remaining bits = $32 - 12 = 20$

Revision - Reducing Memory Overheads of Paging

- 32 bit address space with 4 KB pages
- 4 KB pages \rightarrow 12 bits?
- Remaining bits = $32 - 12 = 20$
 - 20 bit VPN

Revision - Reducing Memory Overheads of Paging

- 32 bit address space with 4 KB pages
- 4 KB pages \rightarrow 12 bits?
- Remaining bits = $32 - 12 = 20$
 - 20 bit VPN
 - # pages = 2^{20}

Revision - Reducing Memory Overheads of Paging

- 32 bit address space with 4 KB pages
- 4 KB pages -> 12 bits?
- Remaining bits = $32 - 12 = 20$
 - 20 bit VPN
 - # pages = 2^{20}
- 4 bytes per translation -> $4 * 2^{20}$ MB = **4 MB/process**

Revision - Reducing Memory Overheads of Paging

Revision - Reducing Memory Overheads of Paging

Solution 0: Decrease the size of VA space

Revision - Reducing Memory Overheads of Paging

Solution 0: Decrease the size of VA space

- 12 bit offset for 4 K pages

Revision - Reducing Memory Overheads of Paging

Solution 0: Decrease the size of VA space

- 12 bit offset for 4 K pages
- 30 bit address space

Revision - Reducing Memory Overheads of Paging

Solution 0: Decrease the size of VA space

- 12 bit offset for 4 K pages
- 30 bit address space
- 18 bit VPNs

Revision - Reducing Memory Overheads of Paging

Solution 0: Decrease the size of VA space

- 12 bit offset for 4 K pages
- 30 bit address space
- 18 bit VPNs
- 4 bytes per translation $\rightarrow 4 * 2^{18} \text{ MB} = \mathbf{1 \text{ MB/process}}$

Revision - Reducing Memory Overheads of Paging

Solution 0: Decrease the size of VA space

- 12 bit offset for 4 K pages
- 30 bit address space
- 18 bit VPNs
- 4 bytes per translation $\rightarrow 4 * 2^{18} \text{ MB} = \mathbf{1 \text{ MB/process}}$

- 32 bit address space $\rightarrow 4 \text{ GB}$

Revision - Reducing Memory Overheads of Paging

Solution 0: Decrease the size of VA space

- 12 bit offset for 4 K pages
- 30 bit address space
- 18 bit VPNs
- 4 bytes per translation $\rightarrow 4 * 2^{18} \text{ MB} = \mathbf{1 \text{ MB/process}}$

- 32 bit address space $\rightarrow 4 \text{ GB}$
- 30 bit address space $\rightarrow 1 \text{ GB}$

Revision - Reducing Memory Overheads of Paging

Revision - Reducing Memory Overheads of Paging

Solution 1: Increase the page size

Revision - Reducing Memory Overheads of Paging

Solution 1: Increase the page size

- 32 bit address space with 16 KB pages

Revision - Reducing Memory Overheads of Paging

Solution 1: Increase the page size

- 32 bit address space with 16 KB pages
- 16 KB pages -> 14 bits?

Revision - Reducing Memory Overheads of Paging

Solution 1: Increase the page size

- 32 bit address space with 16 KB pages
- 16 KB pages \rightarrow 14 bits?
- Remaining bits = $32 - 14 = 18$

Revision - Reducing Memory Overheads of Paging

Solution 1: Increase the page size

- 32 bit address space with 16 KB pages
- 16 KB pages \rightarrow 14 bits?
- Remaining bits = $32 - 14 = 18$
 - 18 bit VPN

Revision - Reducing Memory Overheads of Paging

Solution 1: Increase the page size

- 32 bit address space with 16 KB pages
- 16 KB pages \rightarrow 14 bits?
- Remaining bits = $32 - 14 = 18$
 - 18 bit VPN
 - # pages = 2^{18}

Revision - Reducing Memory Overheads of Paging

Solution 1: Increase the page size

- 32 bit address space with 16 KB pages
- 16 KB pages \rightarrow 14 bits?
- Remaining bits = $32 - 14 = 18$
 - 18 bit VPN
 - # pages = 2^{18}
- 4 bytes per translation $\rightarrow 4 * 2^{18}$ MB = **1 MB/process**

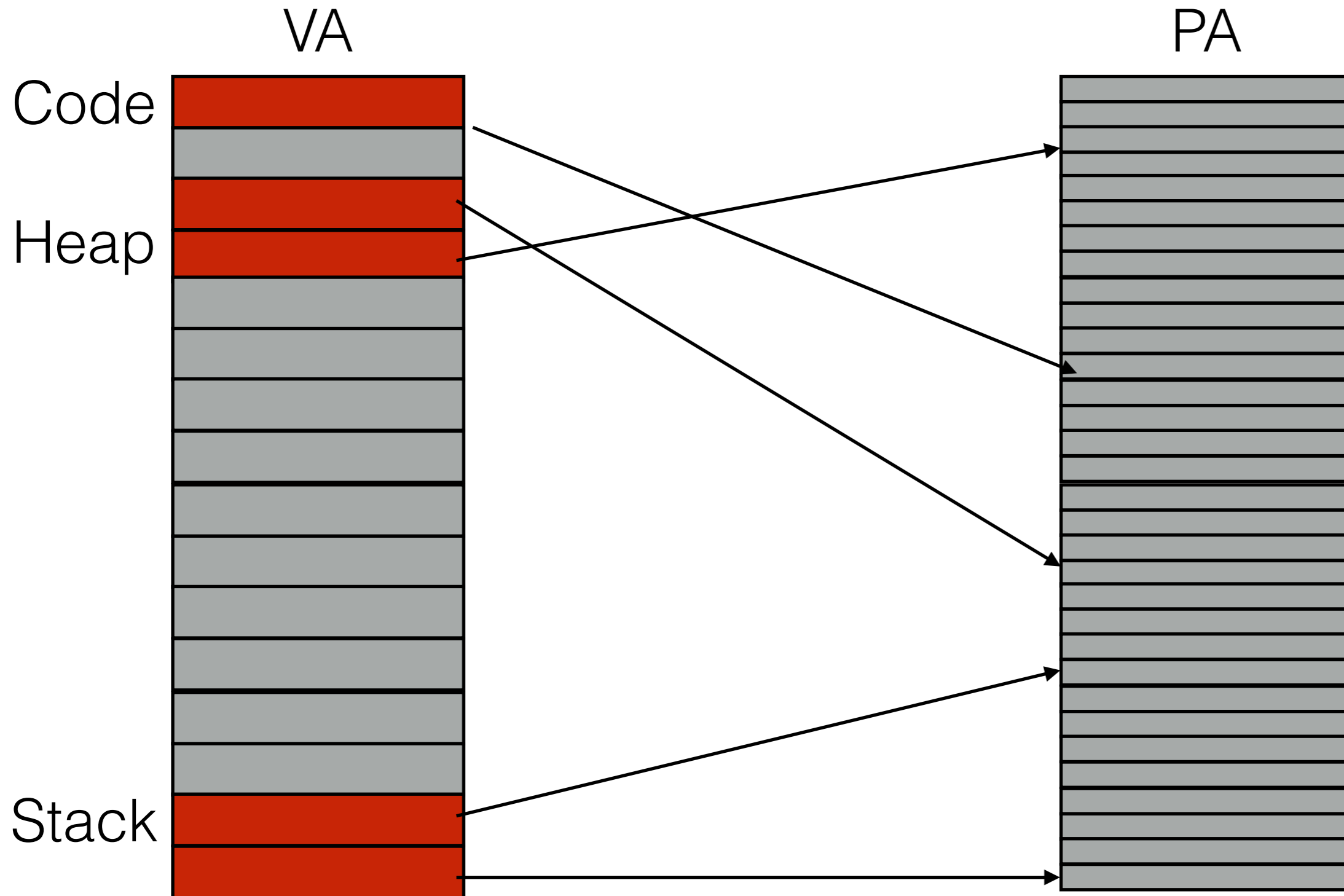
Revision - Reducing Memory Overheads of Paging

Solution 1: Increase the page size

- 32 bit address space with 16 KB pages
- 16 KB pages \rightarrow 14 bits?
- Remaining bits = $32 - 14 = 18$
 - 18 bit VPN
 - # pages = 2^{18}
- 4 bytes per translation $\rightarrow 4 * 2^{18}$ MB = **1 MB/process**

Larger page size \rightarrow Fragmentation


Reducing Memory Overheads of Paging



Reducing Memory Overheads of Paging

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Reducing Memory Overheads of Paging



PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

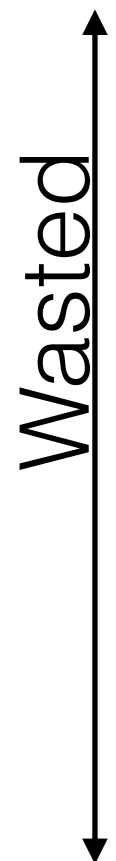
Reducing Memory Overheads of Paging

	PFN	Valid	...
	10	1	..
	-	0	
	-	0	
	-	0	
	23	1	
	-	0	
	-	0	
	-	0	
	-	0	
	-	0	
	-	0	
	-	0	
	-	0	
	-	0	
	28	1	
	4	1	

Wasted

Reducing Memory Overheads of Paging

Wasted



PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Reducing Memory Overheads of Paging

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Linear Page Table

Wasted

Reducing Memory Overheads of Paging

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Linear Page Table

Wasted

Reducing Memory Overheads of Paging

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Wasted



Linear Page Table

Lookup = $O(1)$

Reducing Memory Overheads of Paging

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Wasted

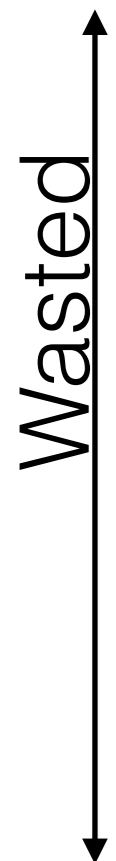


Linear Page Table

Lookup = $O(1)$
Space = $16 * \text{Size}$

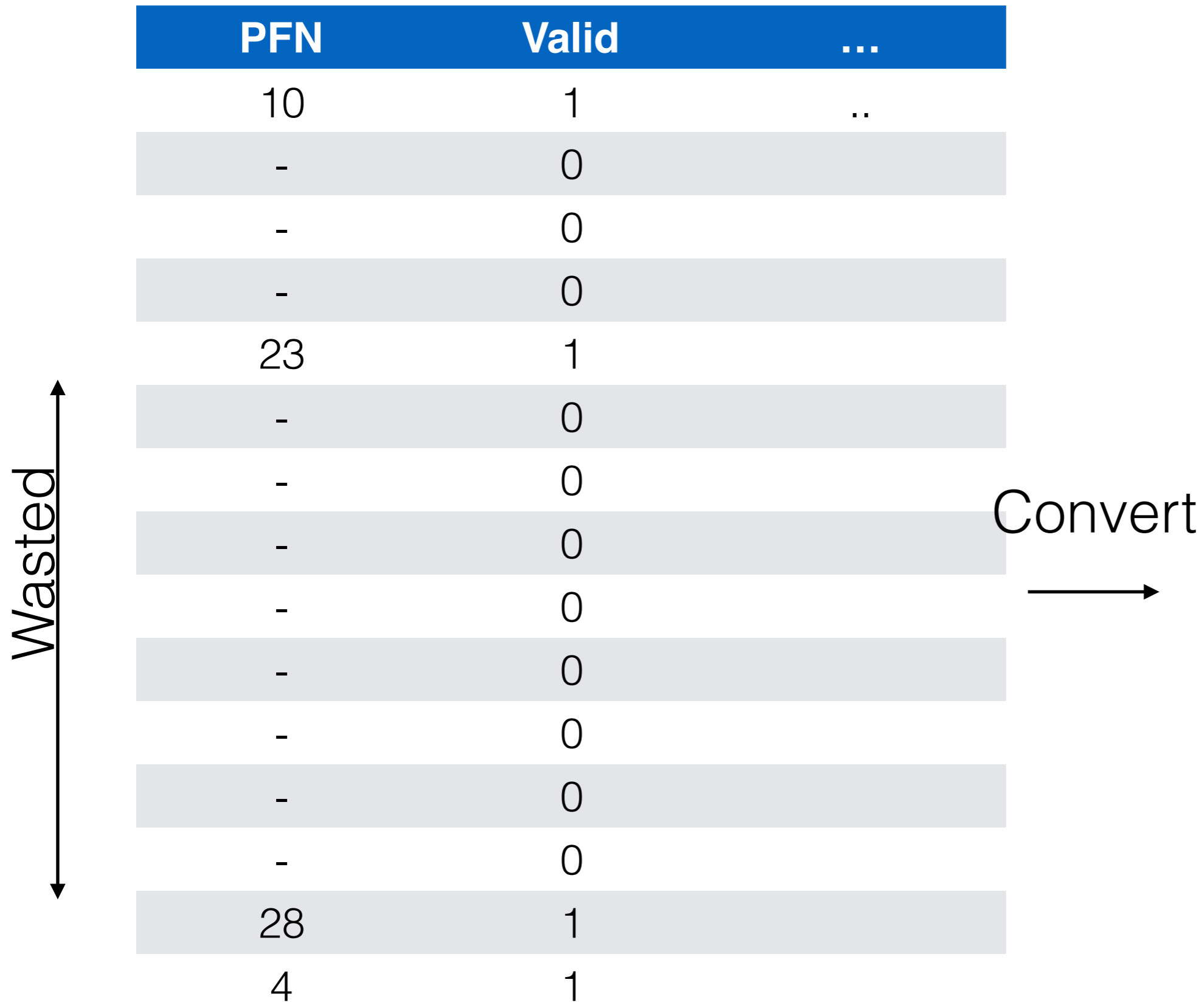
Reducing Memory Overheads of Paging

Wasted

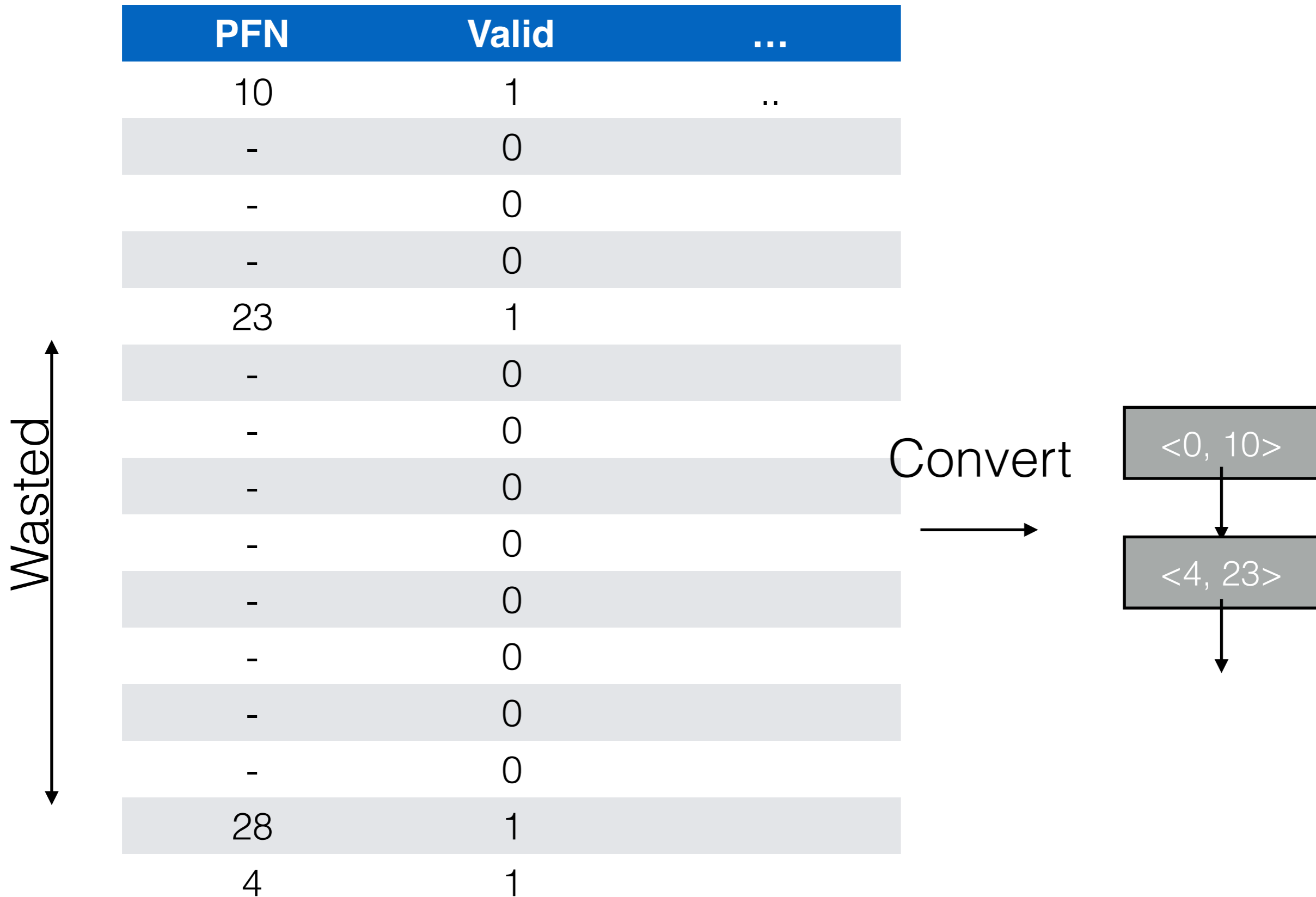


PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

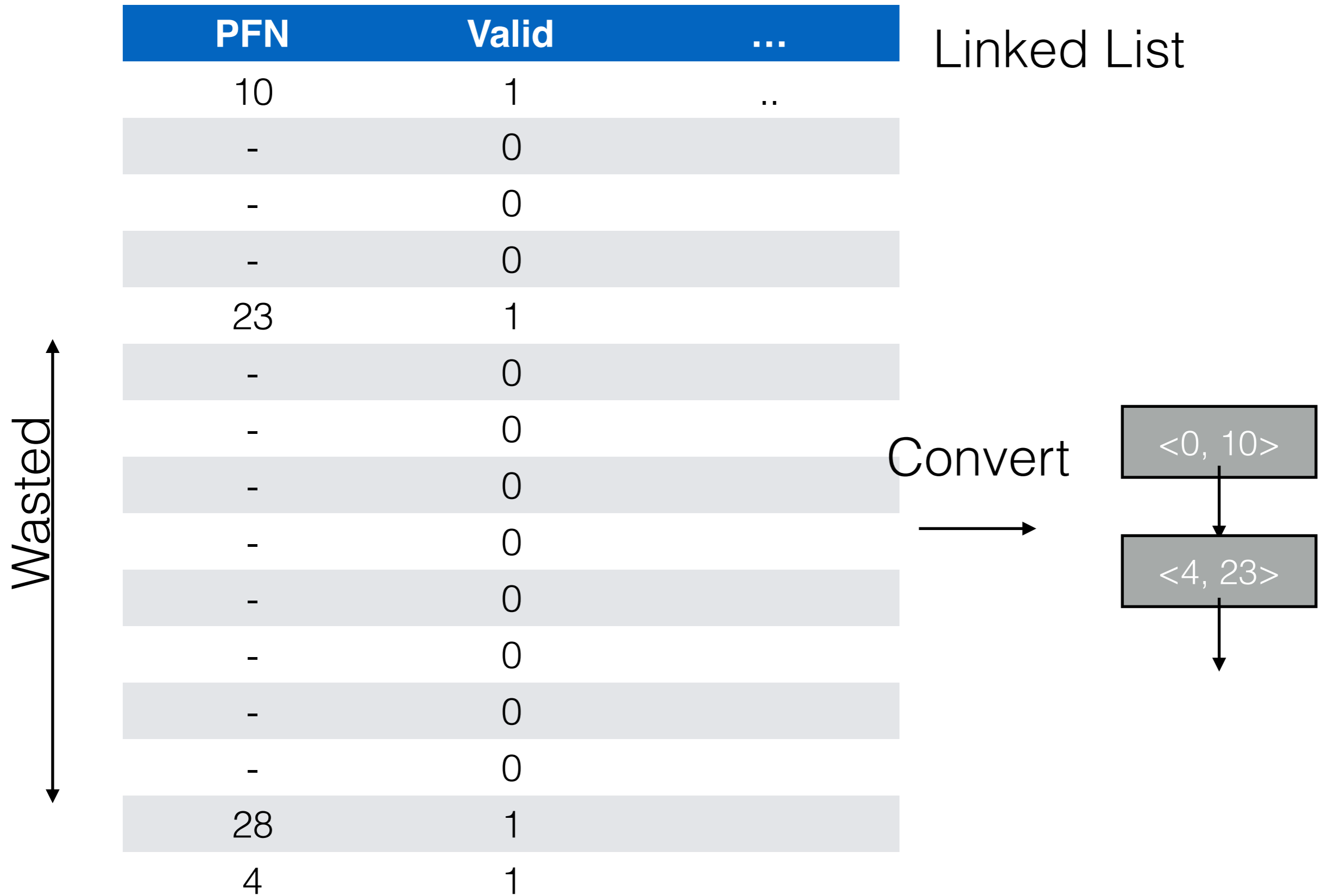
Reducing Memory Overheads of Paging



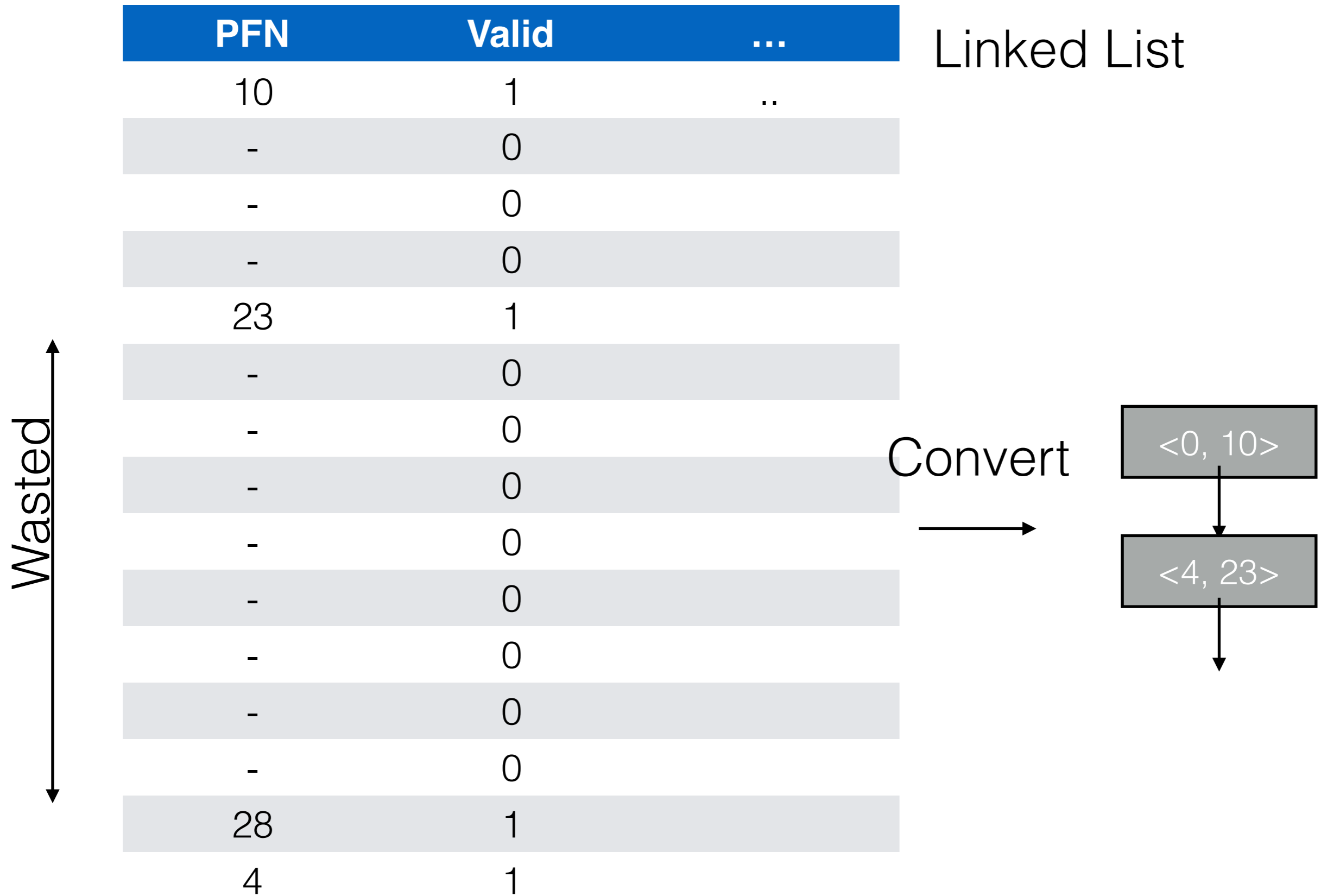
Reducing Memory Overheads of Paging



Reducing Memory Overheads of Paging



Reducing Memory Overheads of Paging



Reducing Memory Overheads of Paging

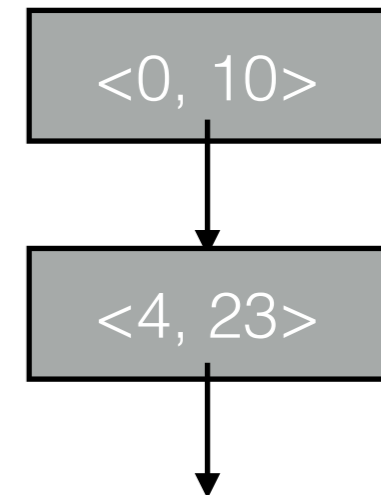
PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Linked List

Lookup = $O(n)$ for n

Wasted

Convert



Reducing Memory Overheads of Paging

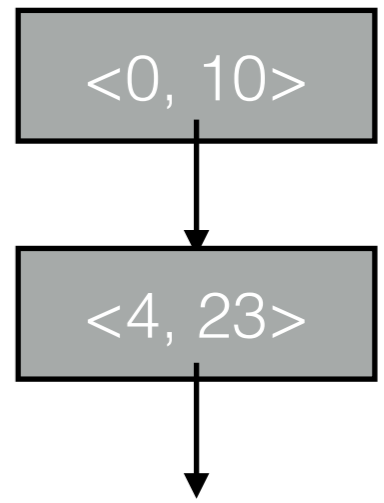
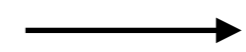
PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Wasted

Linked List

Lookup = $O(n)$ for n valid pages ($n \ll 16$)

Convert



Reducing Memory Overheads of Paging

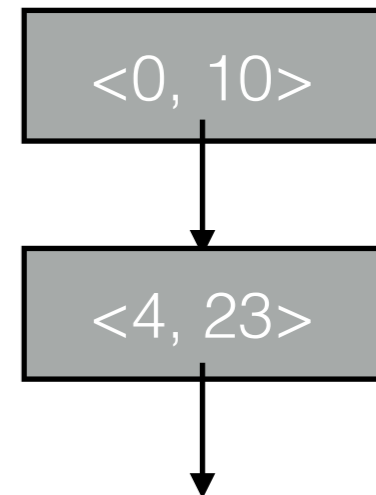
PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Wasted

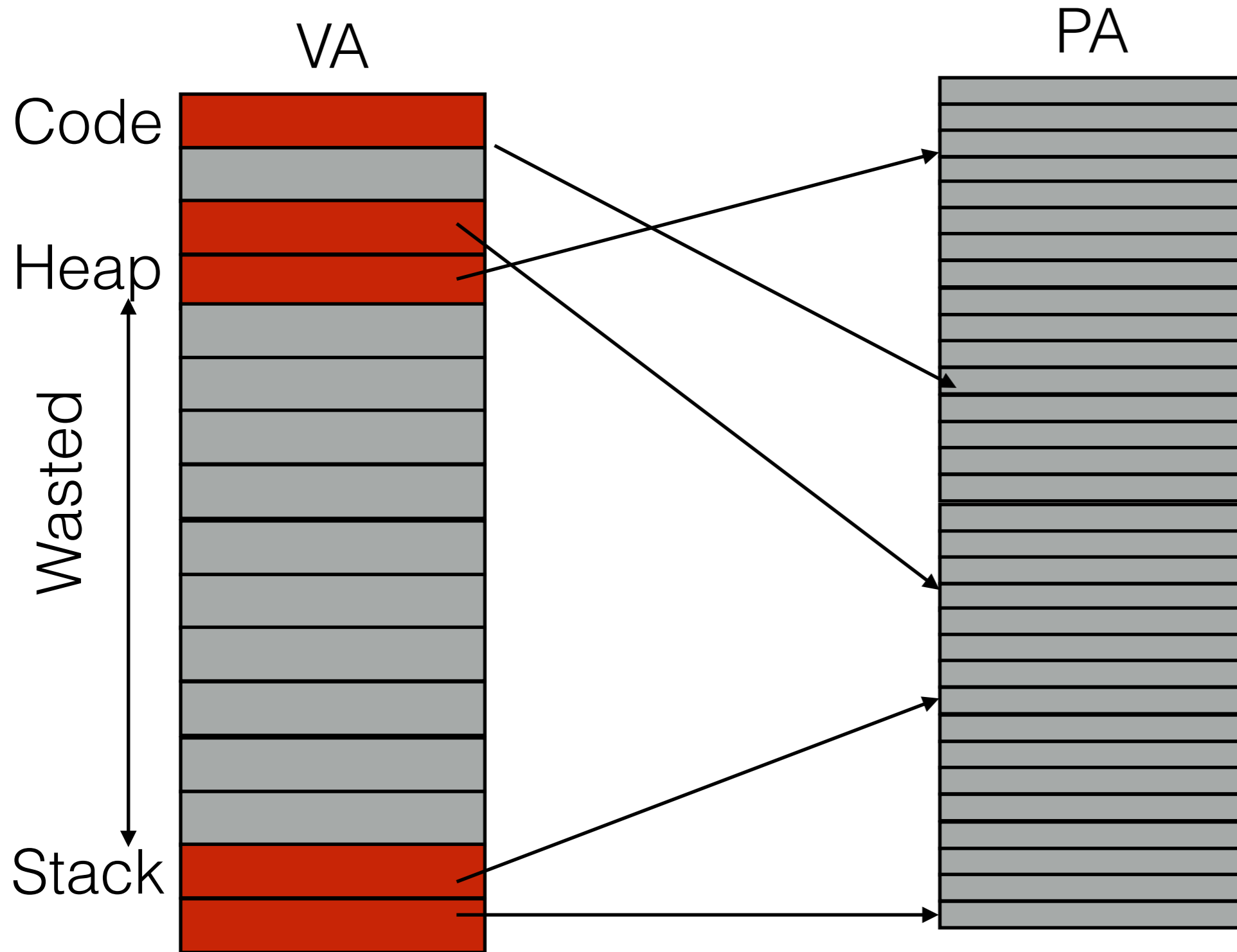
Linked List

Lookup = $O(n)$ for n valid pages ($n \ll 16$)
 Space = $n * \text{Size}$

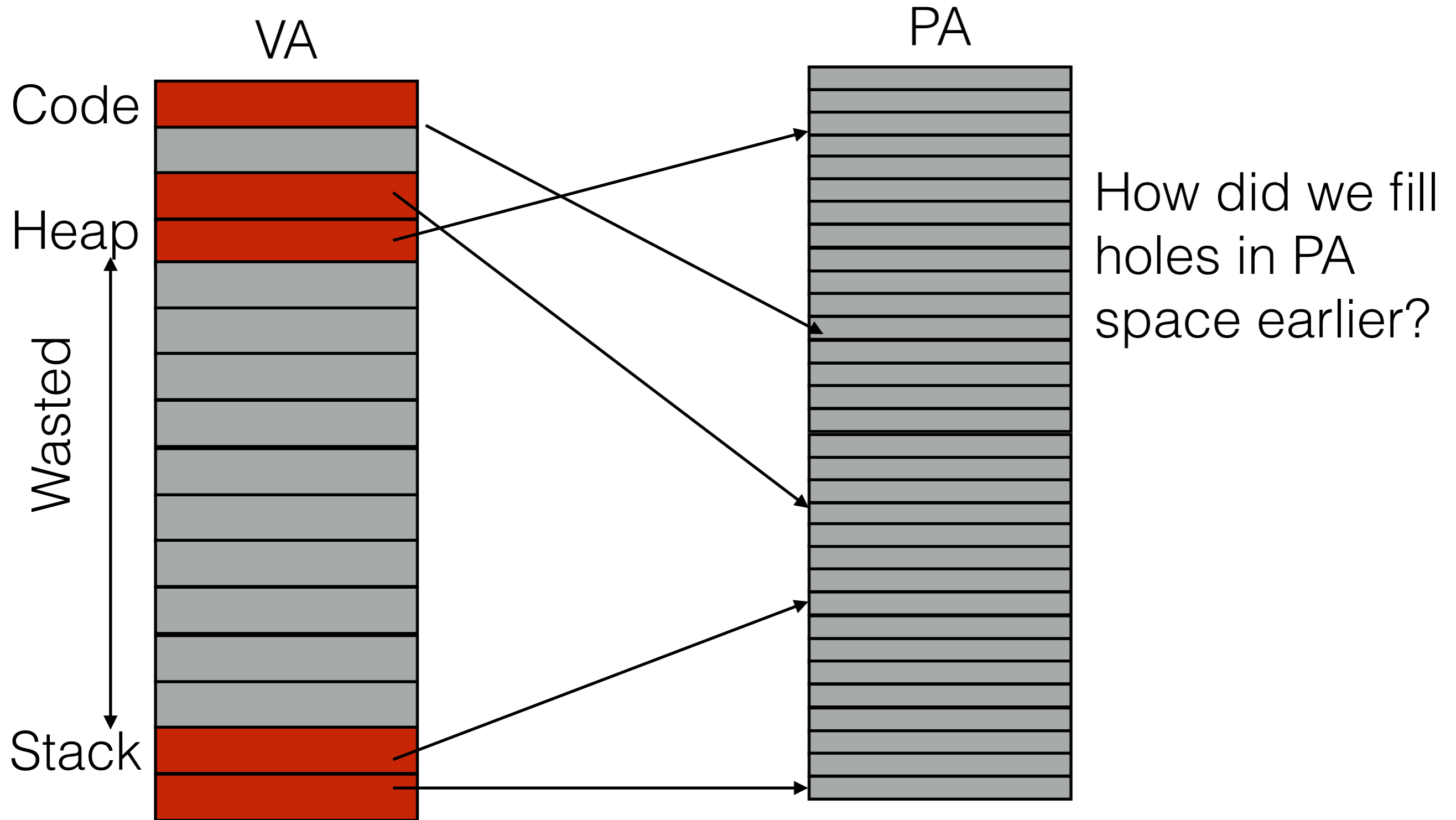
Convert



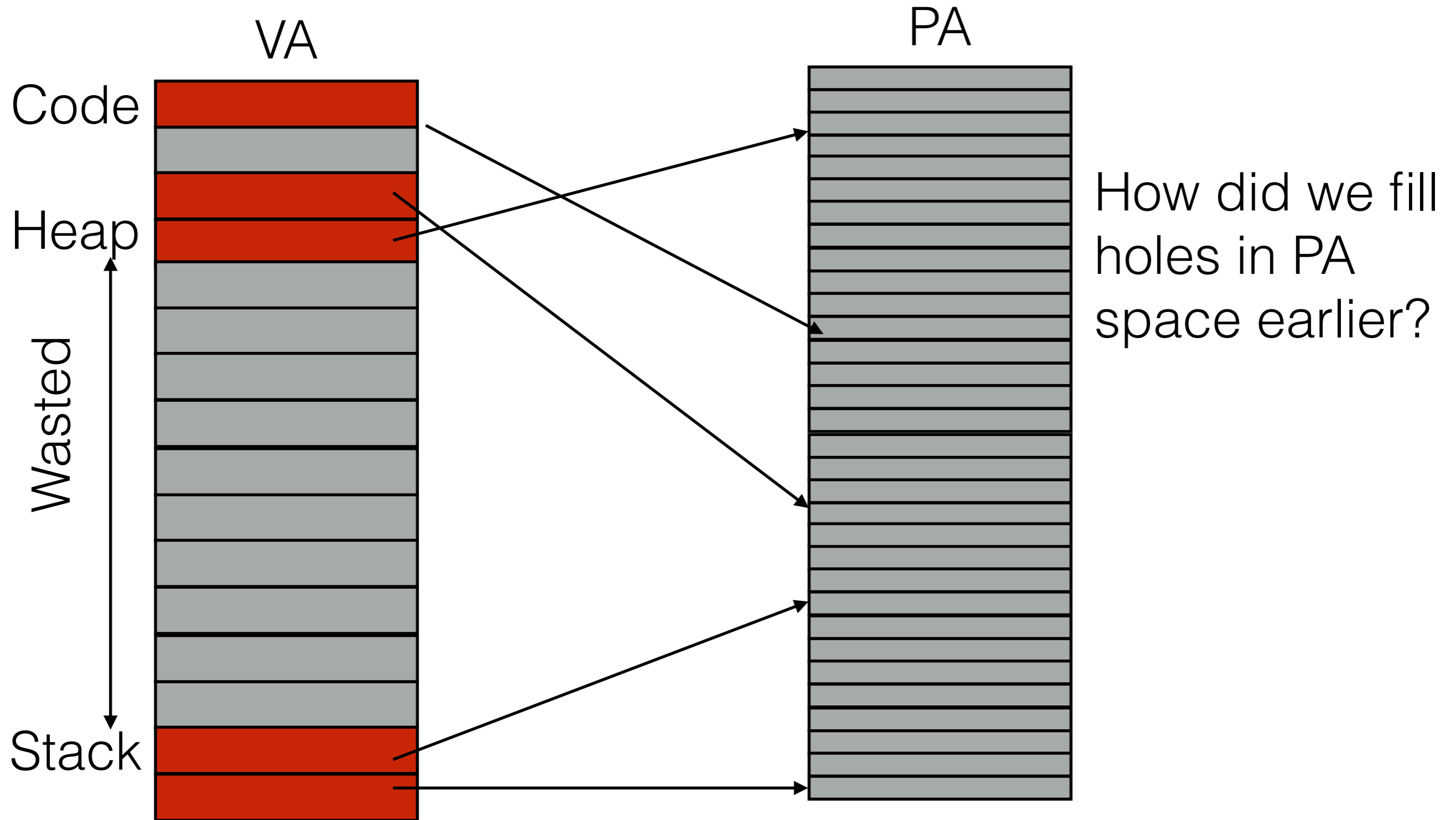
Reducing Memory Overheads of Paging



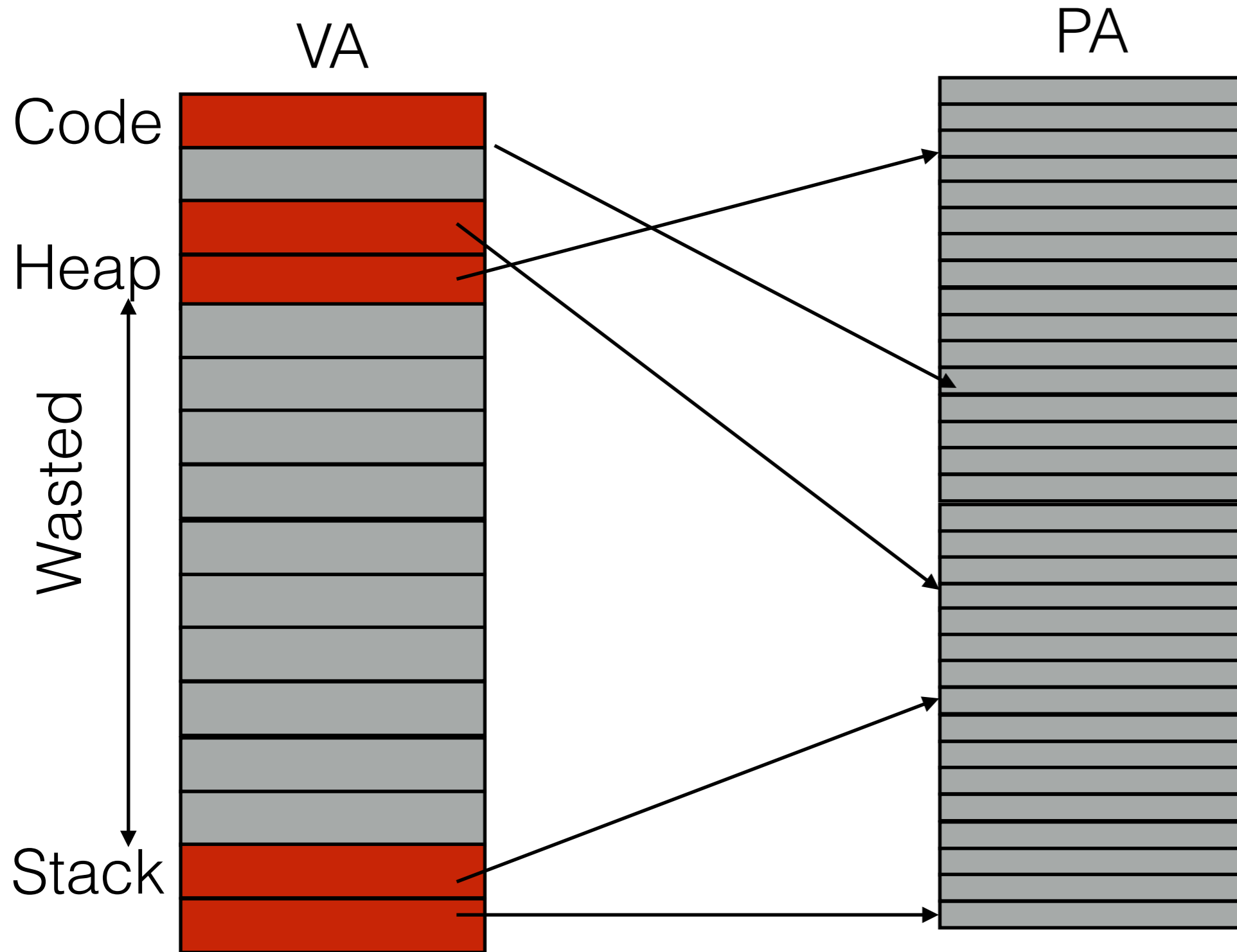
Reducing Memory Overheads of Paging



Reducing Memory Overheads of Paging



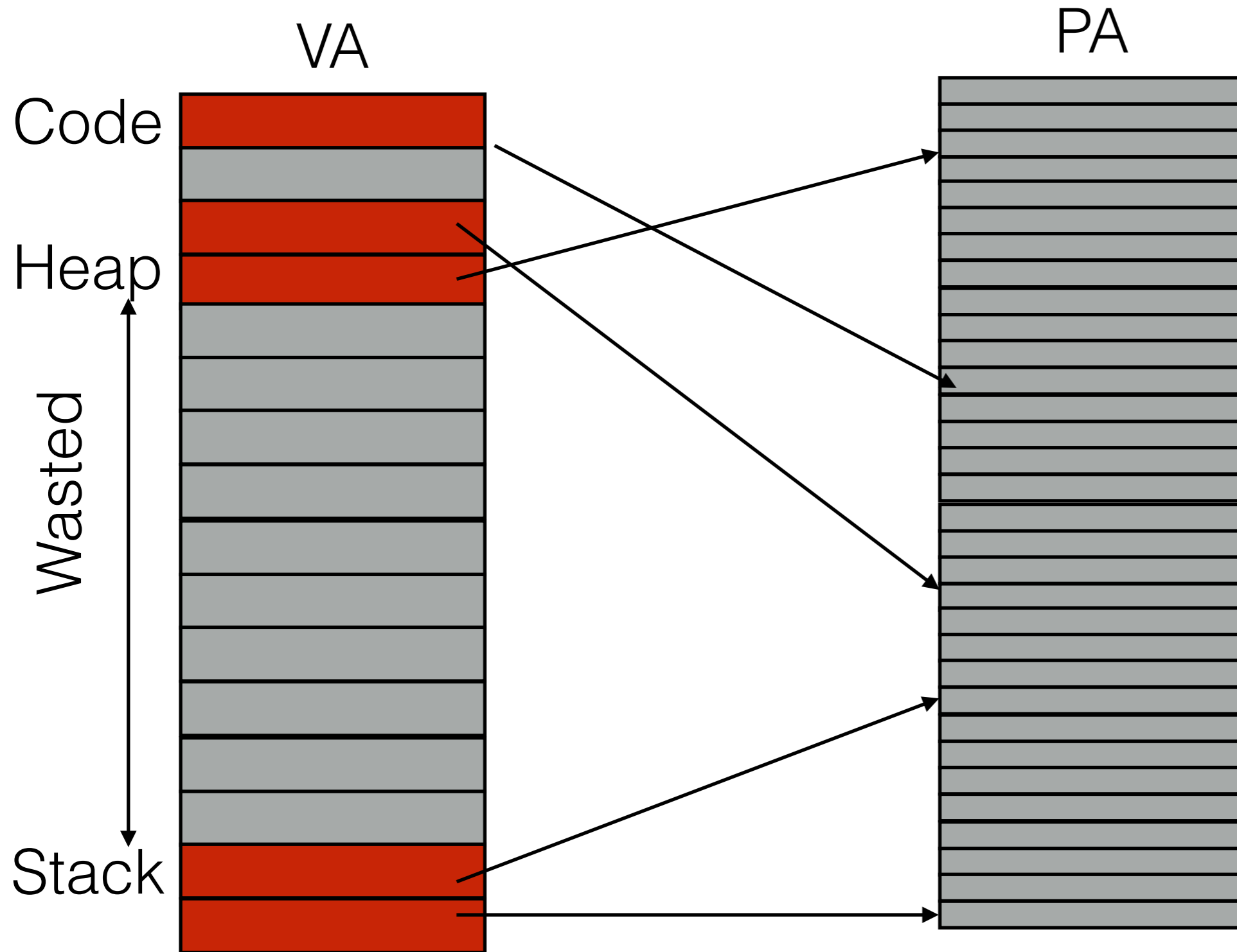
Reducing Memory Overheads of Paging



How did we fill holes in PA space earlier?

1. Segmentation

Reducing Memory Overheads of Paging



How did we fill holes in PA space earlier?

1. Segmentation
2. Paging

Reducing Memory Overheads of Paging - Segmentation + PT

Reducing Memory Overheads of Paging - Segmentation + PT

Idea: use different page tables for heap, stack, etc

Reducing Memory Overheads of Paging - Segmentation + PT

Idea: use different page tables for heap, stack, etc

Reducing Memory Overheads of Paging - Segmentation + PT

Idea: use different page tables for heap, stack, etc

- Each PT can be different size

Reducing Memory Overheads of Paging - Segmentation + PT

Idea: use different page tables for heap, stack, etc

- Each PT can be different size
- Each PT has base & bounds

Reducing Memory Overheads of Paging - Segmentation + PT

Idea: use different page tables for heap, stack, etc

- Each PT can be different size
- Each PT has base & bounds
 - Base & Bounds stored in :

Reducing Memory Overheads of Paging - Segmentation + PT

Idea: use different page tables for heap, stack, etc

- Each PT can be different size
- Each PT has base & bounds
 - Base & Bounds stored in :
 - MMU

Reducing Memory Overheads of Paging - Segmentation + PT

Idea: use different page tables for heap, stack, etc

- Each PT can be different size
- Each PT has base & bounds
 - Base & Bounds stored in :
 - MMU
 - What did Base store in regular segmentation?

Reducing Memory Overheads of Paging - Segmentation + PT

Idea: use different page tables for heap, stack, etc

- Each PT can be different size
- Each PT has base & bounds
 - Base & Bounds stored in :
 - MMU
 - What did Base store in regular segmentation?
 - PA where segment resided

Reducing Memory Overheads of Paging - Segmentation + PT

Idea: use different page tables for heap, stack, etc

- Each PT can be different size
- Each PT has base & bounds
 - Base & Bounds stored in :
 - MMU
 - What did Base store in regular segmentation?
 - PA where segment resided
- What would Base refer here?

Reducing Memory Overheads of Paging - Segmentation + PT

Idea: use different page tables for heap, stack, etc

- Each PT can be different size
- Each PT has base & bounds
 - Base & Bounds stored in :
 - MMU
 - What did Base store in regular segmentation?
 - PA where segment resided
 - What would Base refer here?
 - PA of PT for segment

Reducing Memory Overheads of Paging - Segmentation + PT

Idea: use different page tables for heap, stack, etc

- Each PT can be different size
- Each PT has base & bounds
 - Base & Bounds stored in :
 - MMU
 - What did Base store in regular segmentation?
 - PA where segment resided
 - What would Base refer here?
 - PA of PT for segment
 - What would Bounds refer to here?

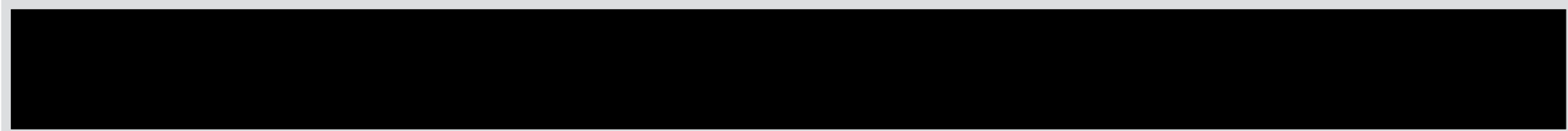
Reducing Memory Overheads of Paging - Segmentation + PT

Idea: use different page tables for heap, stack, etc

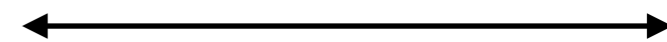
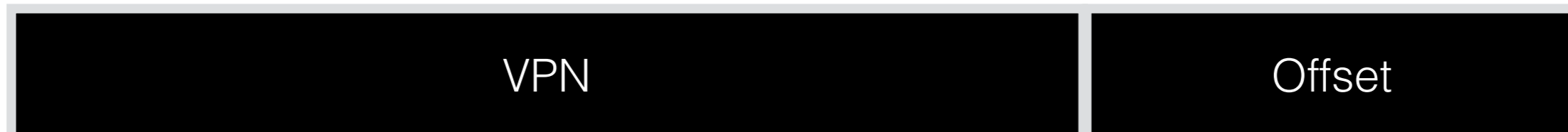
- Each PT can be different size
- Each PT has base & bounds
 - Base & Bounds stored in :
 - MMU
 - What did Base store in regular segmentation?
 - PA where segment resided
 - What would Base refer here?
 - PA of PT for segment
 - What would Bounds refer to here?
 - Number of valid pages/End of page table

Reducing Memory Overheads of Paging - Segmentation + PT

32 bit VA space

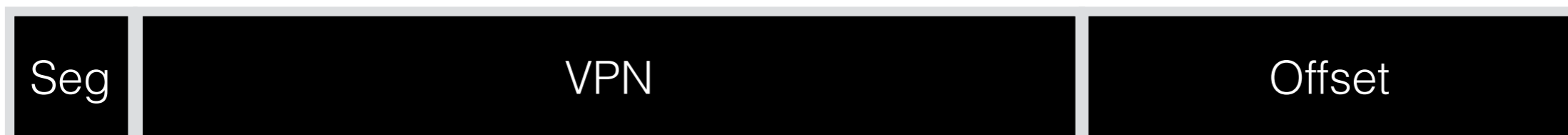


32 bit VA space with 4KB pages



12 bits

32 bit VA space with 4KB pages for 4 segments



2 bits

Reducing Memory Overheads of Paging - Segmentation + PT

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Reducing Memory Overheads of Paging - Segmentation + PT

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

PFN	Valid	...
10	1	..
..	1	

Reducing Memory Overheads of Paging - Segmentation + PT

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Code PT

PFN	Valid	...
10	1	..
..	1	

Reducing Memory Overheads of Paging - Segmentation + PT

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Code PT
Base = 0

PFN	Valid	...
10	1	..
..	1	

Reducing Memory Overheads of Paging - Segmentation + PT

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Code PT
Base = 0
Bounds = 2

PFN	Valid	...
10	1	..
..	1	

Reducing Memory Overheads of Paging - Segmentation + PT

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Code PT
Base = 0
Bounds = 2

PFN	Valid	...
10	1	..
..	1	

PFN	Valid	...
23	1	..
..	1	

Reducing Memory Overheads of Paging - Segmentation + PT

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Code PT
Base = 0
Bounds = 2

PFN	Valid	...
10	1	..
..	1	

Heap PT

PFN	Valid	...
23	1	..
..	1	

Reducing Memory Overheads of Paging - Segmentation + PT

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Code PT
Base = 0
Bounds = 2

PFN	Valid	...
10	1	..
..	1	

Heap PT
Base = 4

PFN	Valid	...
23	1	..
..	1	

Reducing Memory Overheads of Paging - Segmentation + PT

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Code PT
Base = 0
Bounds = 2

PFN	Valid	...
10	1	..
..	1	

Heap PT
Base = 4
Bounds = 2

PFN	Valid	...
23	1	..
..	1	

Reducing Memory Overheads of Paging - Segmentation + PT

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Code PT
Base = 0
Bounds = 2

PFN	Valid	...
10	1	..
..	1	

Heap PT
Base = 4
Bounds = 2

PFN	Valid	...
23	1	..
..	1	

Stack PT

Reducing Memory Overheads of Paging - Segmentation + PT

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Code PT
Base = 0
Bounds = 2

PFN	Valid	...
10	1	..
..	1	

Heap PT
Base = 4
Bounds = 2

PFN	Valid	...
23	1	..
..	1	

Stack PT
Base = ?

Reducing Memory Overheads of Paging - Segmentation + PT

PFN	Valid	...
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Code PT
Base = 0
Bounds = 2

PFN	Valid	...
10	1	..
..	1	

Heap PT
Base = 4
Bounds = 2

PFN	Valid	...
23	1	..
..	1	

Stack PT
Base = ?
Bounds = ?

Reducing Memory Overheads of Paging - Segmentation + PT

Reducing Memory Overheads of Paging - Segmentation + PT

Reducing Memory Overheads of Paging - Segmentation + PT

- Pros:

Reducing Memory Overheads of Paging - Segmentation + PT

- Pros:
 - Leads to memory saving (Large gaps between segments)

Reducing Memory Overheads of Paging - Segmentation + PT

- Pros:
 - Leads to memory saving (Large gaps between segments)
- Cons:

Reducing Memory Overheads of Paging - Segmentation + PT

- Pros:
 - Leads to memory saving (Large gaps between segments)
- Cons:
 - Uses segmentation

Reducing Memory Overheads of Paging - Segmentation + PT

- Pros:
 - Leads to memory saving (Large gaps between segments)
- Cons:
 - Uses segmentation
 - Assumes certain usage pattern of address space

Reducing Memory Overheads of Paging - Segmentation + PT

- Pros:
 - Leads to memory saving (Large gaps between segments)
- Cons:
 - Uses segmentation
 - Assumes certain usage pattern of address space
 - Sparsely used segments (sub-segment internal fragmentation) have same space waste issue

Reducing Memory Overheads of Paging - Segmentation + PT

- Pros:
 - Leads to memory saving (Large gaps between segments)
- Cons:
 - Uses segmentation
 - Assumes certain usage pattern of address space
 - Sparsely used segments (sub-segment internal fragmentation) have same space waste issue
 - How to address this?

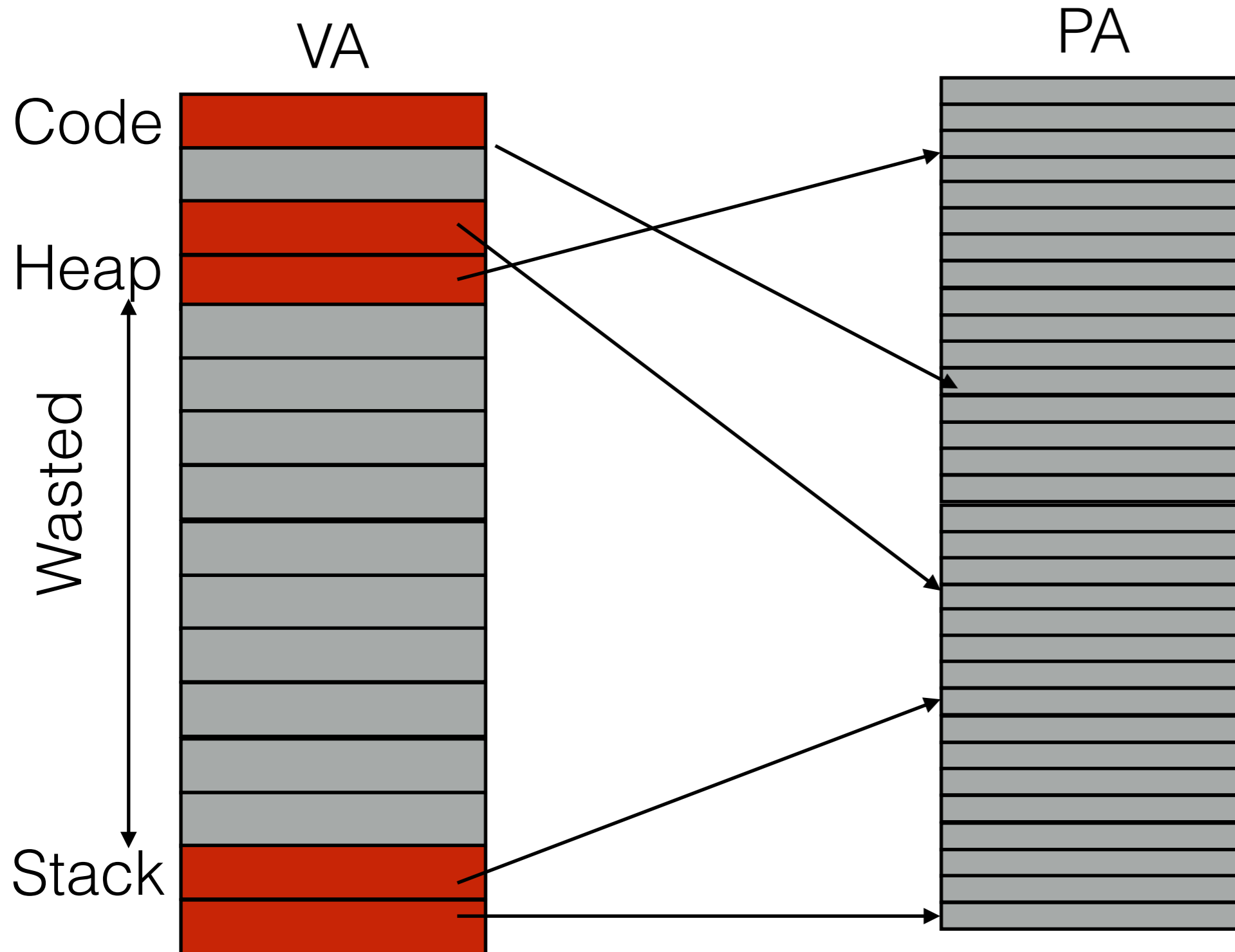
Reducing Memory Overheads of Paging - Segmentation + PT

- Pros:
 - Leads to memory saving (Large gaps between segments)
- Cons:
 - Uses segmentation
 - Assumes certain usage pattern of address space
 - Sparsely used segments (sub-segment internal fragmentation) have same space waste issue
 - How to address this?
 - **LinkedList! Getting complex now.**

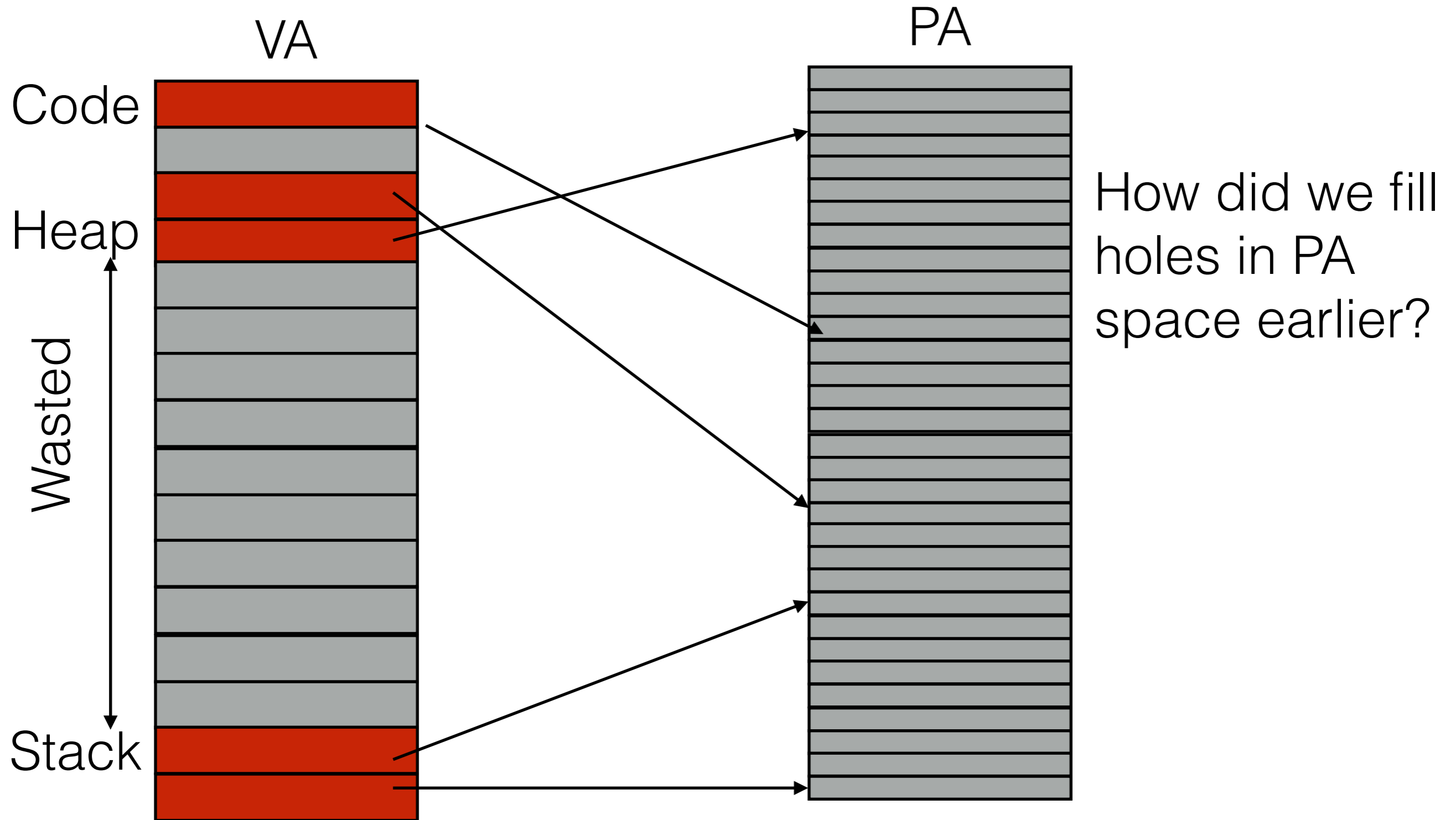
Reducing Memory Overheads of Paging - Segmentation + PT

- Pros:
 - Leads to memory saving (Large gaps between segments)
- Cons:
 - Uses segmentation
 - Assumes certain usage pattern of address space
 - Sparsely used segments (sub-segment internal fragmentation) have same space waste issue
 - How to address this?
 - LinkedList! Getting complex now.
 - **Variable size page tables —> Can lead to fragmentation**

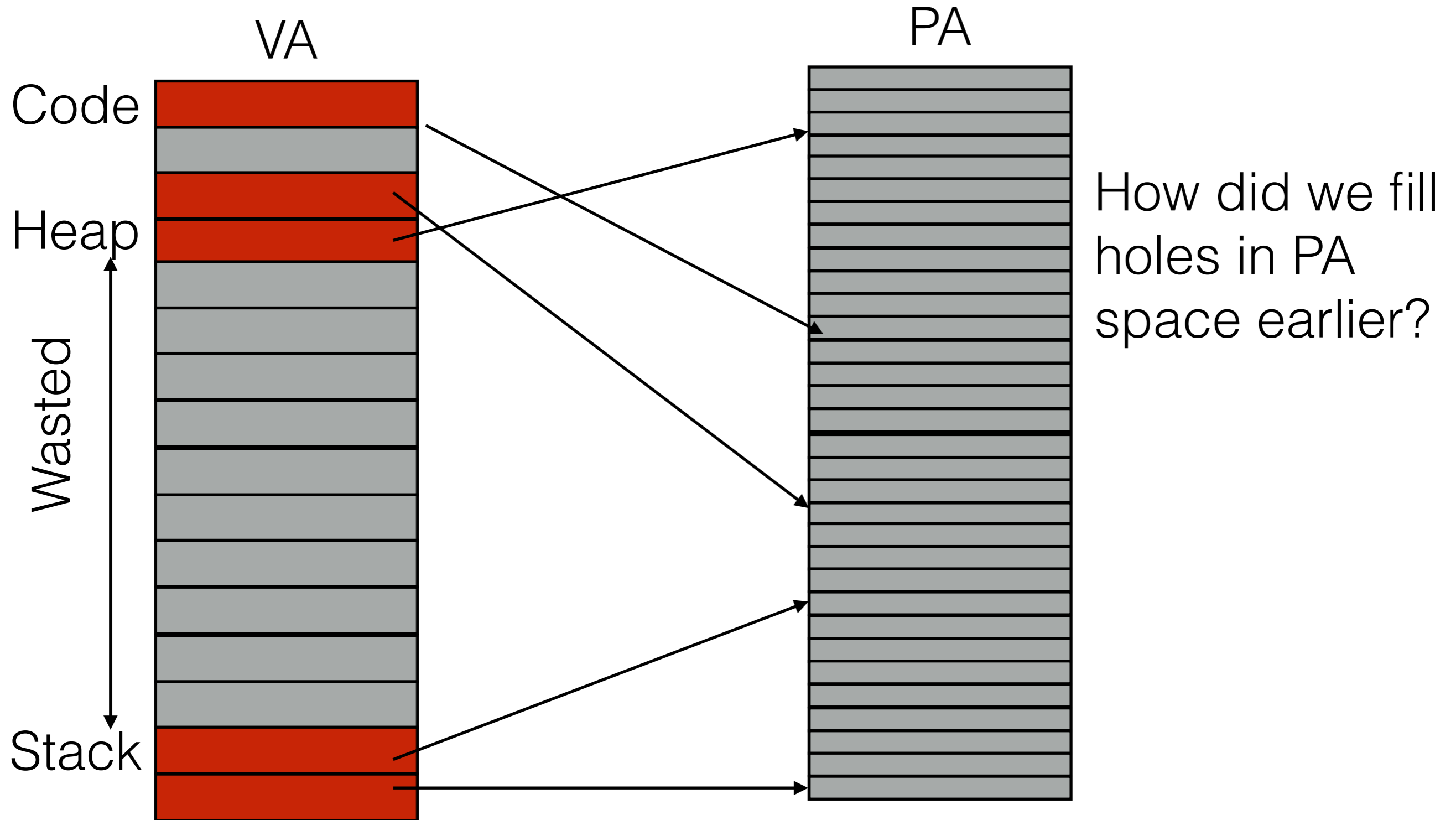
Reducing Memory Overheads of Paging



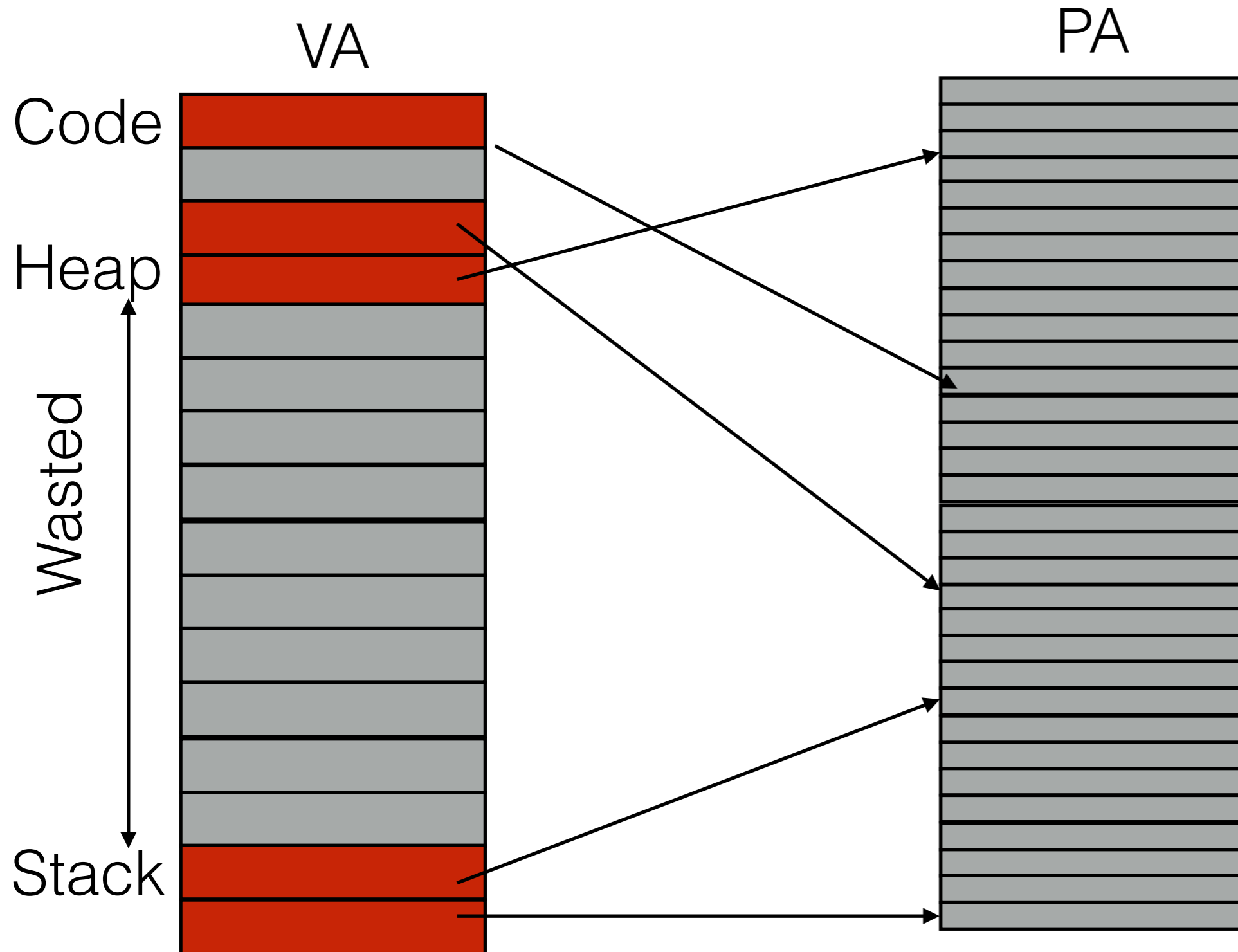
Reducing Memory Overheads of Paging



Reducing Memory Overheads of Paging



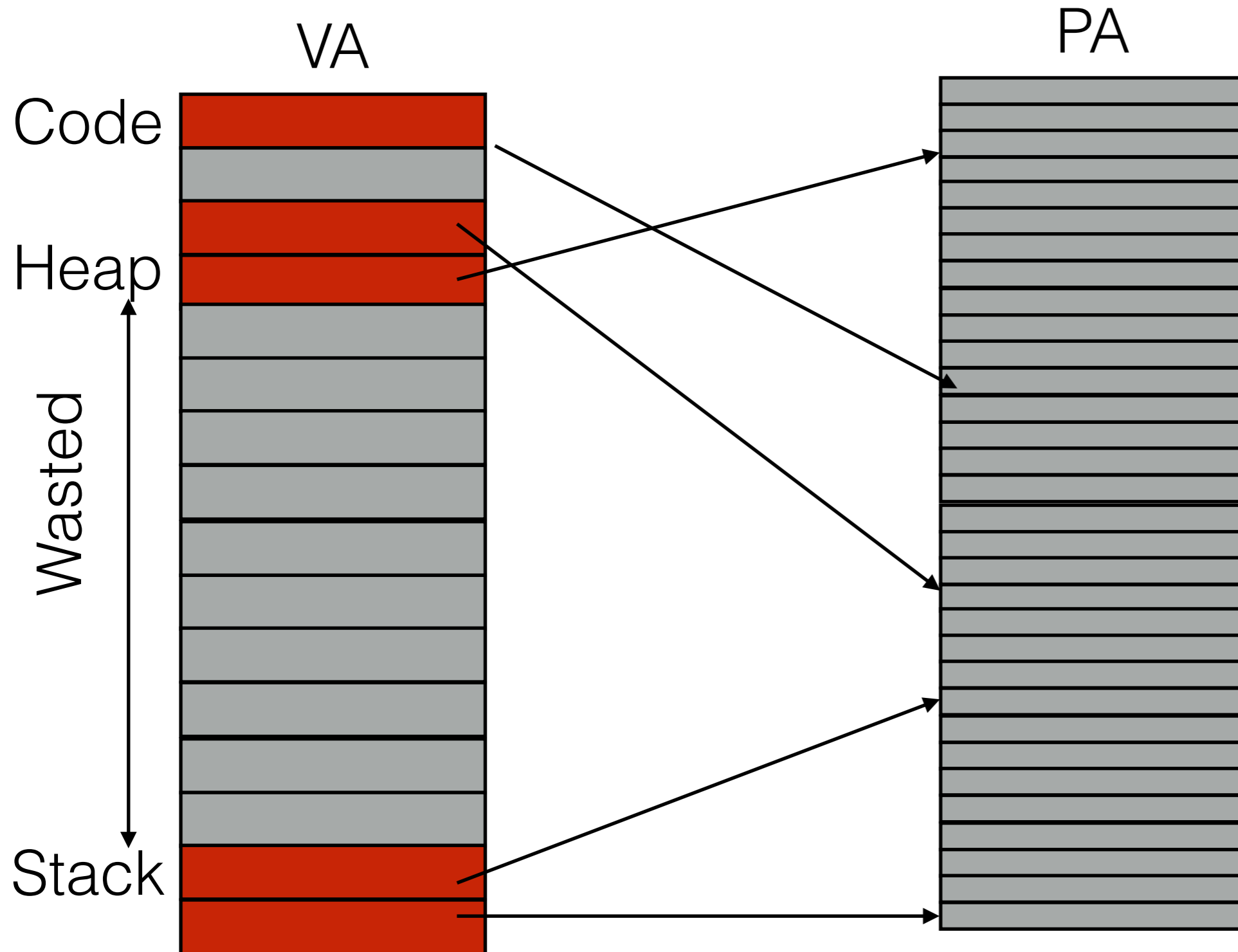
Reducing Memory Overheads of Paging



How did we fill holes in PA space earlier?

1. Segmentation

Reducing Memory Overheads of Paging



How did we fill holes in PA space earlier?

1. Segmentation
- 2. Paging**

Administrative

Administrative

- lab mid sem exam on 16th September (Monday)

Administrative

- lab mid sem exam on 16th September (Monday)
- memory virtualisation quiz on 13th September (next Friday)

Administrative

- lab mid sem exam on 16th September (Monday)
- memory virtualisation quiz on 13th September (next Friday)
- project first phase updates on 25th September or 1-3 days earlier

Administrative

- lab mid sem exam on 16th September (Monday)
- memory virtualisation quiz on 13th September (next Friday)
- project first phase updates on 25th September or 1-3 days earlier
- I'll try to grade their project proposals by tomorrow

Administrative

- lab mid sem exam on 16th September (Monday)
- memory virtualisation quiz on 13th September (next Friday)
- project first phase updates on 25th September or 1-3 days earlier
- I'll try to grade their project proposals by tomorrow
- I'll also put the slides up by tonight

Administrative

- lab mid sem exam on 16th September (Monday)
- memory virtualisation quiz on 13th September (next Friday)
- project first phase updates on 25th September or 1-3 days earlier
- I'll try to grade their project proposals by tomorrow
- I'll also put the slides up by tonight
- Before mid see break: 2 X 10 marks quizzes, 10 marks for lab exam, 4 marks for HW1, and about 8 marks for project; all totaling 42 marks

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging)

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging)

Idea: break PT itself into pages

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging)

Idea: break PT itself into pages

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging)

Idea: break PT itself into pages

- A Page Directory refers to pieces

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging)

Idea: break PT itself into pages

- A Page Directory refers to pieces
- Only have pieces with >0 valid entries

Reducing Memory Overheads of Paging - Segmentation + PT

	PFN	Valid	...
PFN = 200	10	1	..
	-	0	
	-	0	
PFN = 201	23	1	
	-	0	
	-	0	
PFN = 202	-	0	
	-	0	
	-	0	
PFN = 203	-	0	
	-	0	
	28	1	
	4	1	

Total entries = 16

Reducing Memory Overheads of Paging - Segmentation + PT

	PFN	Valid	...
PFN = 200	10	1	..
	-	0	
	-	0	
PFN = 201	23	1	
	-	0	
	-	0	
PFN = 202	-	0	
	-	0	
	-	0	
PFN = 203	-	0	
	-	0	
	28	1	
	4	1	

Total entries = 16

Reducing Memory Overheads of Paging - Segmentation + PT

	PFN	Valid	...
PFN = 200	10	1	..
	-	0	
	-	0	
	-	0	
PFN = 201	23	1	
	-	0	
	-	0	
	-	0	
PFN = 202	-	0	
	-	0	
	-	0	
	-	0	
PFN = 203	-	0	
	-	0	
	28	1	
	4	1	

PFN	Valid	...
10	1	..
-	0	
-	0	
23	1	...

Total entries = 16

Reducing Memory Overheads of Paging - Segmentation + PT

	PFN	Valid	...
PFN = 200	10	1	..
	-	0	
	-	0	
	-	0	
PFN = 201	23	1	
	-	0	
	-	0	
	-	0	
PFN = 202	-	0	
	-	0	
	-	0	
	-	0	
PFN = 203	-	0	
	-	0	
	28	1	
	4	1	

Total entries = 16

PFN	Valid	...
10	1	..
-	0	
-	0	
23	1	...

PFN	Valid	...
-	0	..
-	0	
-	0	
-	0	...

Reducing Memory Overheads of Paging - Segmentation + PT

	PFN	Valid	...
PFN = 200	10	1	..
	-	0	
	-	0	
	-	0	
PFN = 201	23	1	
	-	0	
	-	0	
	-	0	
PFN = 202	-	0	
	-	0	
	-	0	
	-	0	
PFN = 203	-	0	
	-	0	
	28	1	
	4	1	

Total entries = 16

PFN	Valid	...
10	1	..
-	0	
-	0	
23	1	...

PFN	Valid	...
-	0	..
-	0	
-	0	
-	0	...

PFN	Valid	...
-	0	..
-	0	
-	0	
-	0	...

Reducing Memory Overheads of Paging - Segmentation + PT

PFN = 200	PFN	Valid	...
	10	1	..
	-	0	
	-	0	
PFN = 201	23	1	
	-	0	
	-	0	
	-	0	
PFN = 202	-	0	
	-	0	
	-	0	
	-	0	
PFN = 203	-	0	
	-	0	
	28	1	
	4	1	

Total entries = 16

PFN	Valid	...
10	1	..
-	0	
-	0	
23	1	...

PFN	Valid	...
-	0	..
-	0	
-	0	
-	0	...

PFN	Valid	...
-	0	..
-	0	
-	0	
-	0	...

PFN	Valid	...
-	0	..
-	0	
28	1	
4	1	...

Reducing Memory Overheads of Paging - Segmentation + PT

PFN = 200

PFN	Valid	...
-----	-------	-----

10	1	..
----	---	----

-	0	
---	---	--

-	0	
---	---	--

-	0	
---	---	--

PFN = 201

23	1	
----	---	--

-	0	
---	---	--

-	0	
---	---	--

-	0	
---	---	--

-	0	
---	---	--

PFN = 202

-	0	
---	---	--

-	0	
---	---	--

-	0	
---	---	--

PFN = 203

-	0	
---	---	--

28	1	
----	---	--

4	1	
---	---	--

Total entries = 16



PFN	Valid	...
-----	-------	-----

201	1	..
-----	---	----

202	0	
-----	---	--

203	0	
-----	---	--

204	1	...
-----	---	-----

PFN	Valid	...
-----	-------	-----

10	1	..
----	---	----

-	0	
---	---	--

-	0	
---	---	--

23	1	...
----	---	-----

PFN	Valid	...
-----	-------	-----

-	0	..
---	---	----

-	0	
---	---	--

-	0	
---	---	--

-	0	...
---	---	-----

PFN	Valid	...
-----	-------	-----

-	0	..
---	---	----

-	0	
---	---	--

-	0	
---	---	--

-	0	...
---	---	-----

PFN	Valid	...
-----	-------	-----

-	0	..
---	---	----

-	0	
---	---	--

28	1	
----	---	--

4	1	...
---	---	-----

Reducing Memory Overheads of Paging - Segmentation + PT

PFN = 200

PFN	Valid	...
-----	-------	-----

10	1	..
----	---	----

-	0	
---	---	--

-	0	
---	---	--

-	0	
---	---	--

PFN = 201

23	1	
----	---	--

-	0	
---	---	--

-	0	
---	---	--

-	0	
---	---	--

-	0	
---	---	--

-	0	
---	---	--

-	0	
---	---	--

-	0	
---	---	--

-	0	
---	---	--

28	1	
----	---	--

4	1	
---	---	--



PFN = 200

PFN	Valid	...
-----	-------	-----

201	1	..
-----	---	----

202	0	
-----	---	--

203	0	
-----	---	--

204	1	...
-----	---	-----

PFN	Valid	...
-----	-------	-----

10	1	..
----	---	----

-	0	
---	---	--

-	0	
---	---	--

23	1	...
----	---	-----

PFN	Valid	...
-----	-------	-----

-	0	..
---	---	----

-	0	
---	---	--

-	0	
---	---	--

-	0	...
---	---	-----

PFN	Valid	...
-----	-------	-----

-	0	..
---	---	----

-	0	
---	---	--

-	0	
---	---	--

-	0	...
---	---	-----

PFN	Valid	...
-----	-------	-----

-	0	..
---	---	----

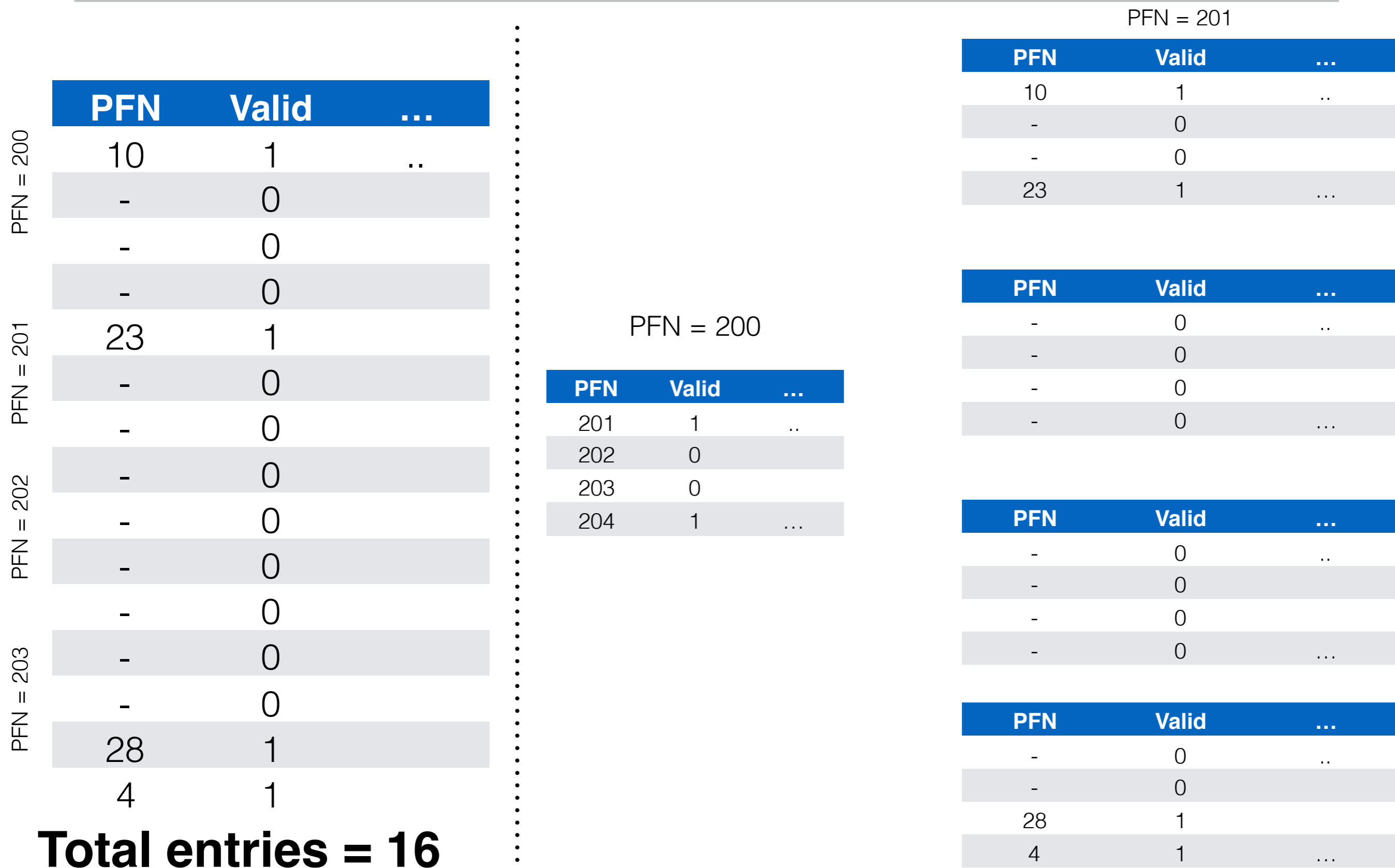
-	0	
---	---	--

28	1	
----	---	--

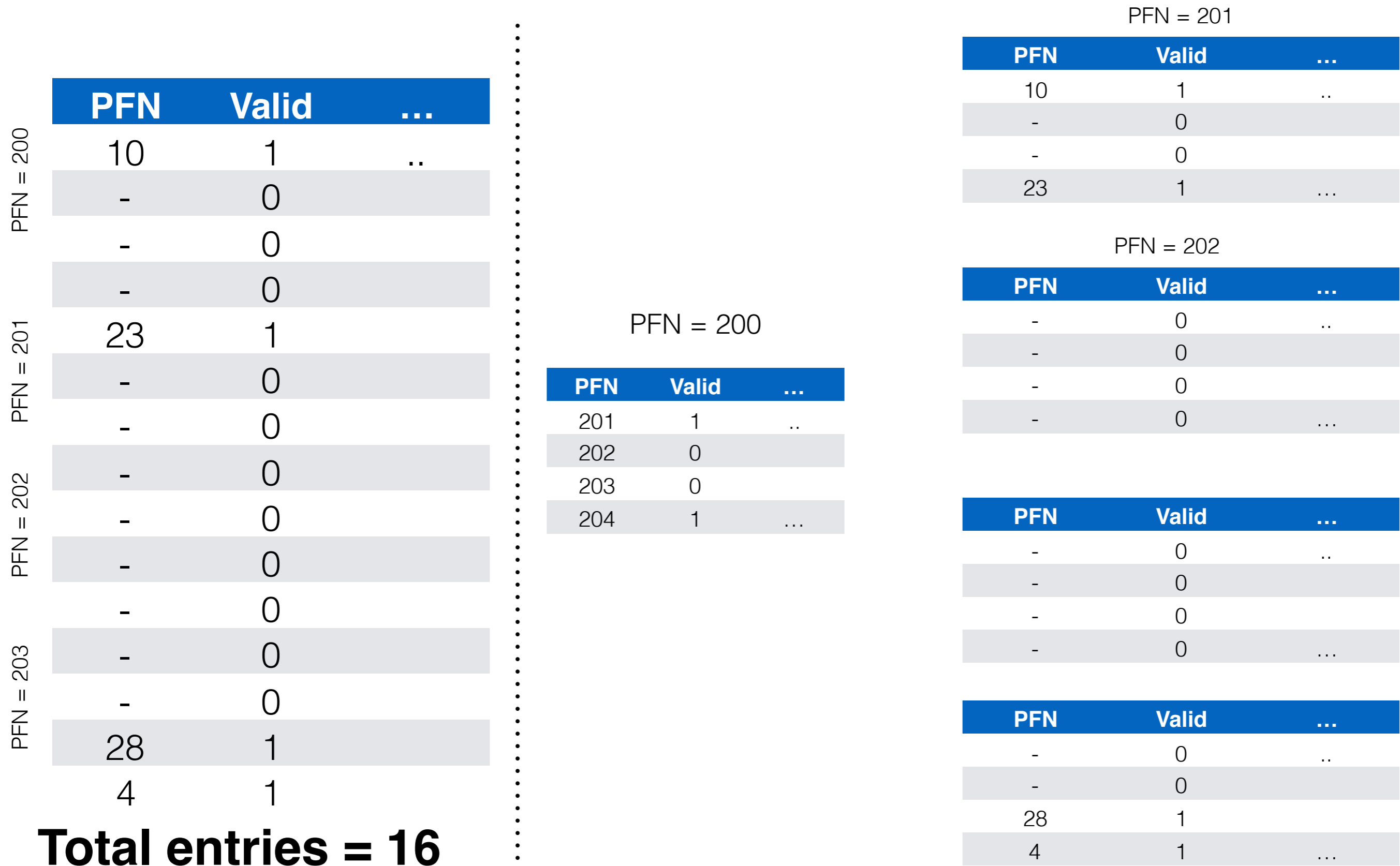
4	1	...
---	---	-----

Total entries = 16

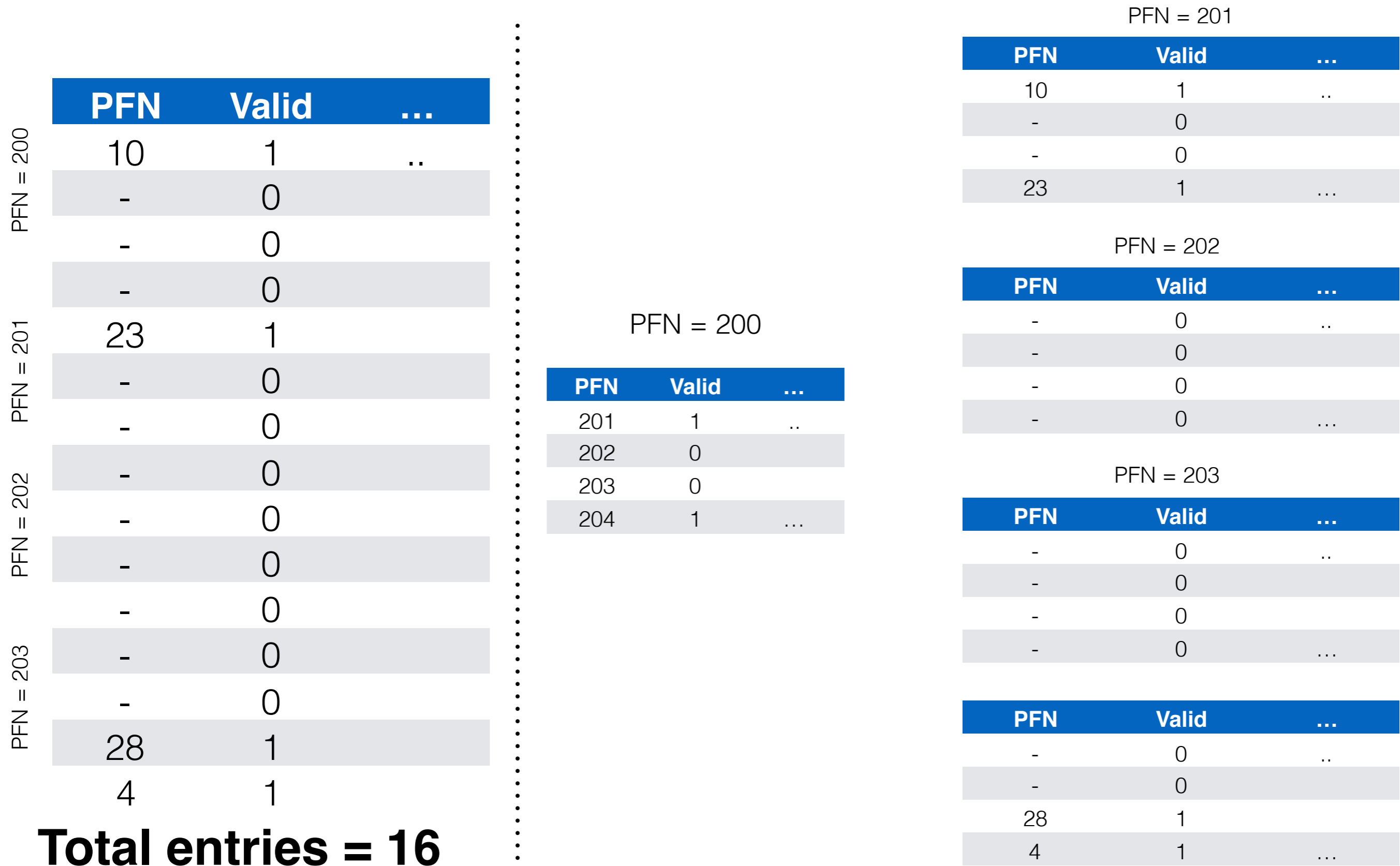
Reducing Memory Overheads of Paging - Segmentation + PT



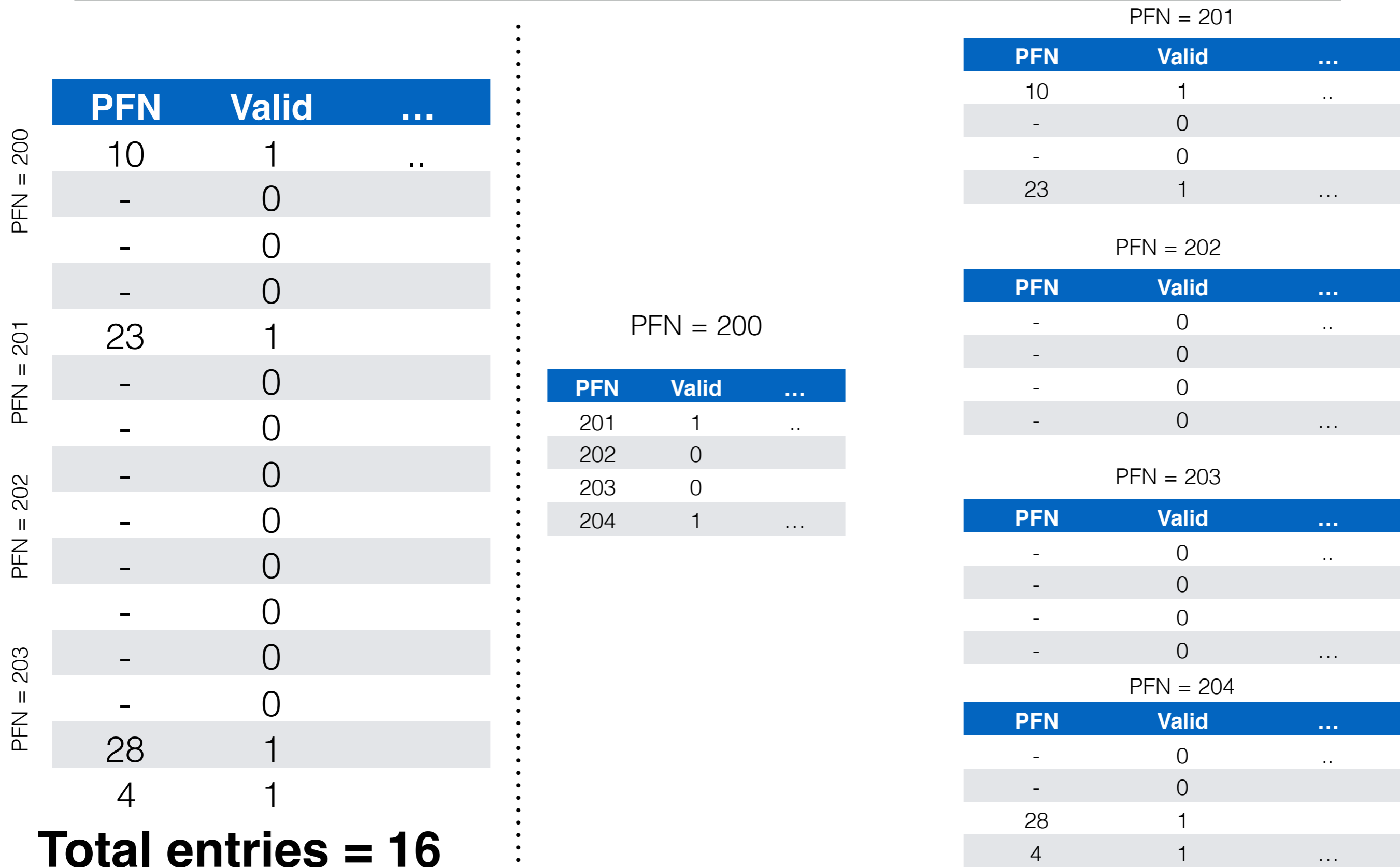
Reducing Memory Overheads of Paging - Segmentation + PT



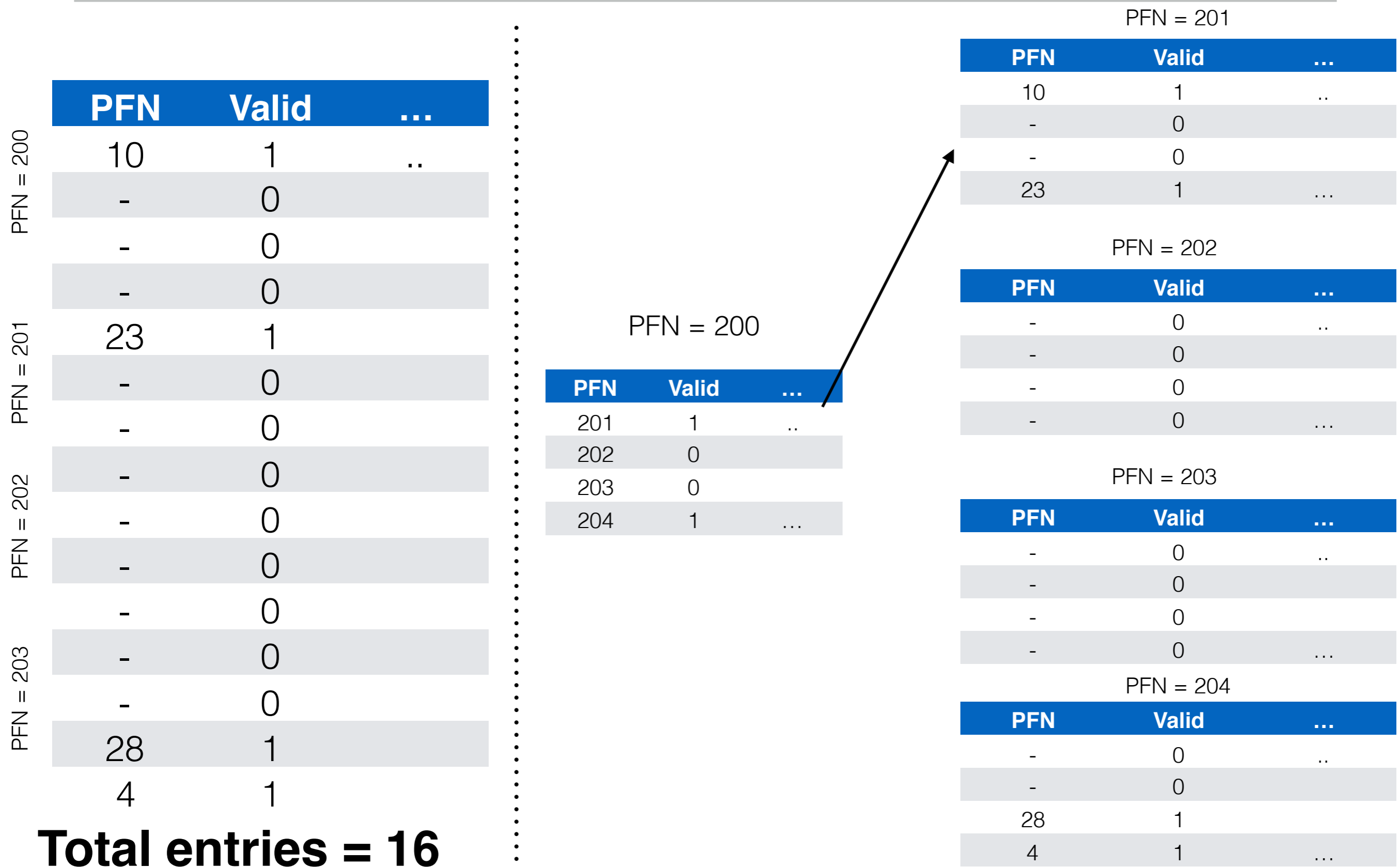
Reducing Memory Overheads of Paging - Segmentation + PT



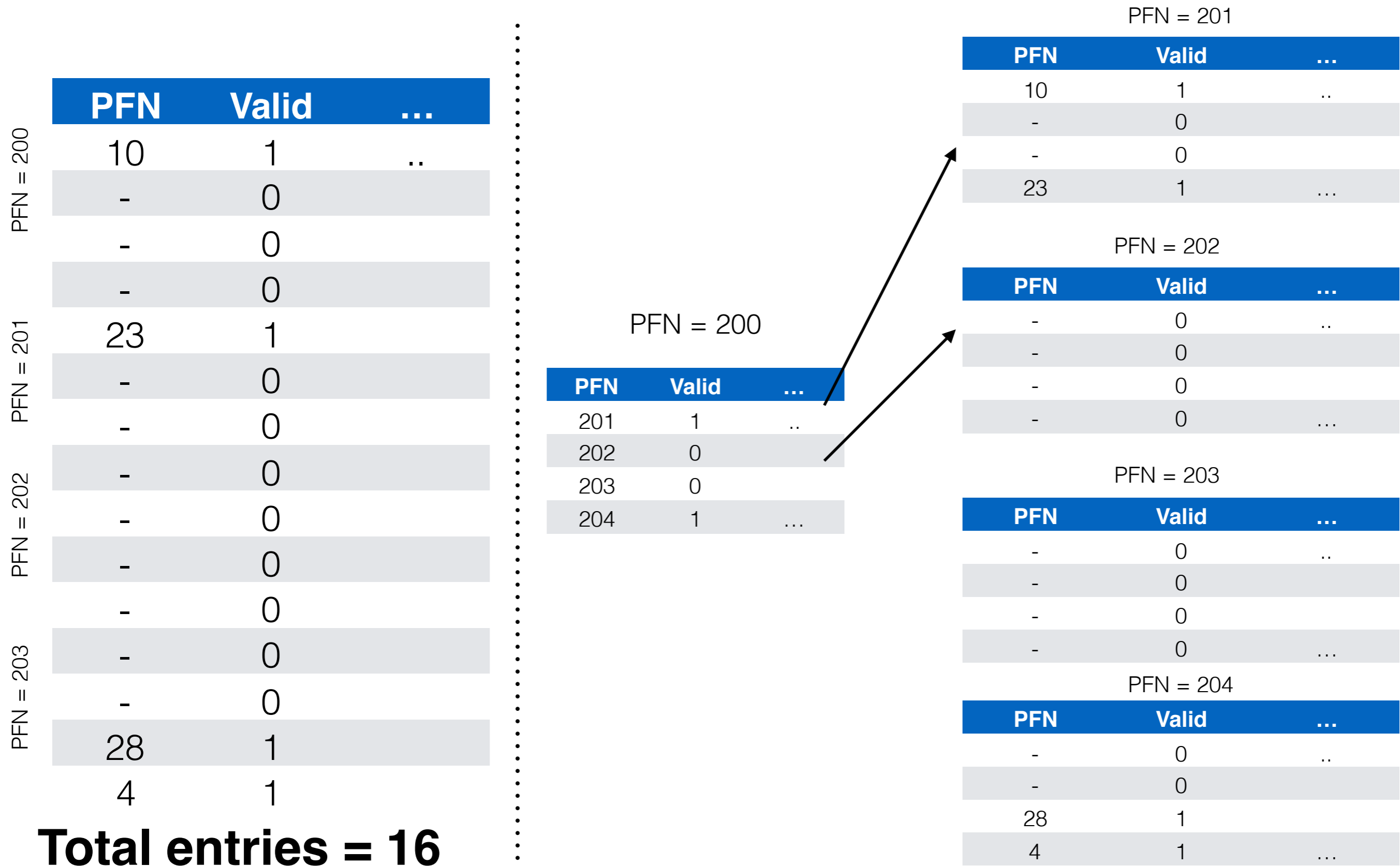
Reducing Memory Overheads of Paging - Segmentation + PT



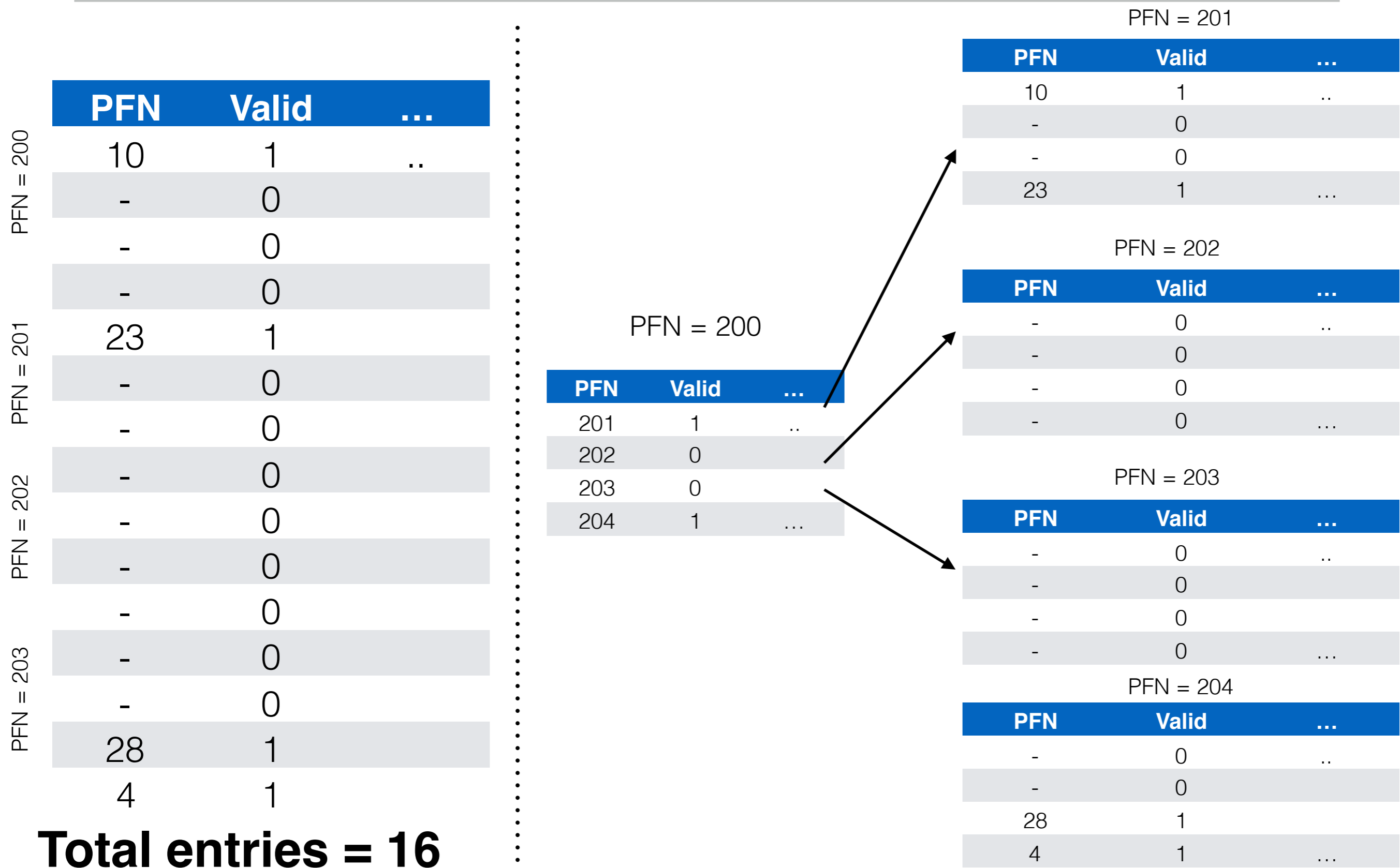
Reducing Memory Overheads of Paging - Segmentation + PT



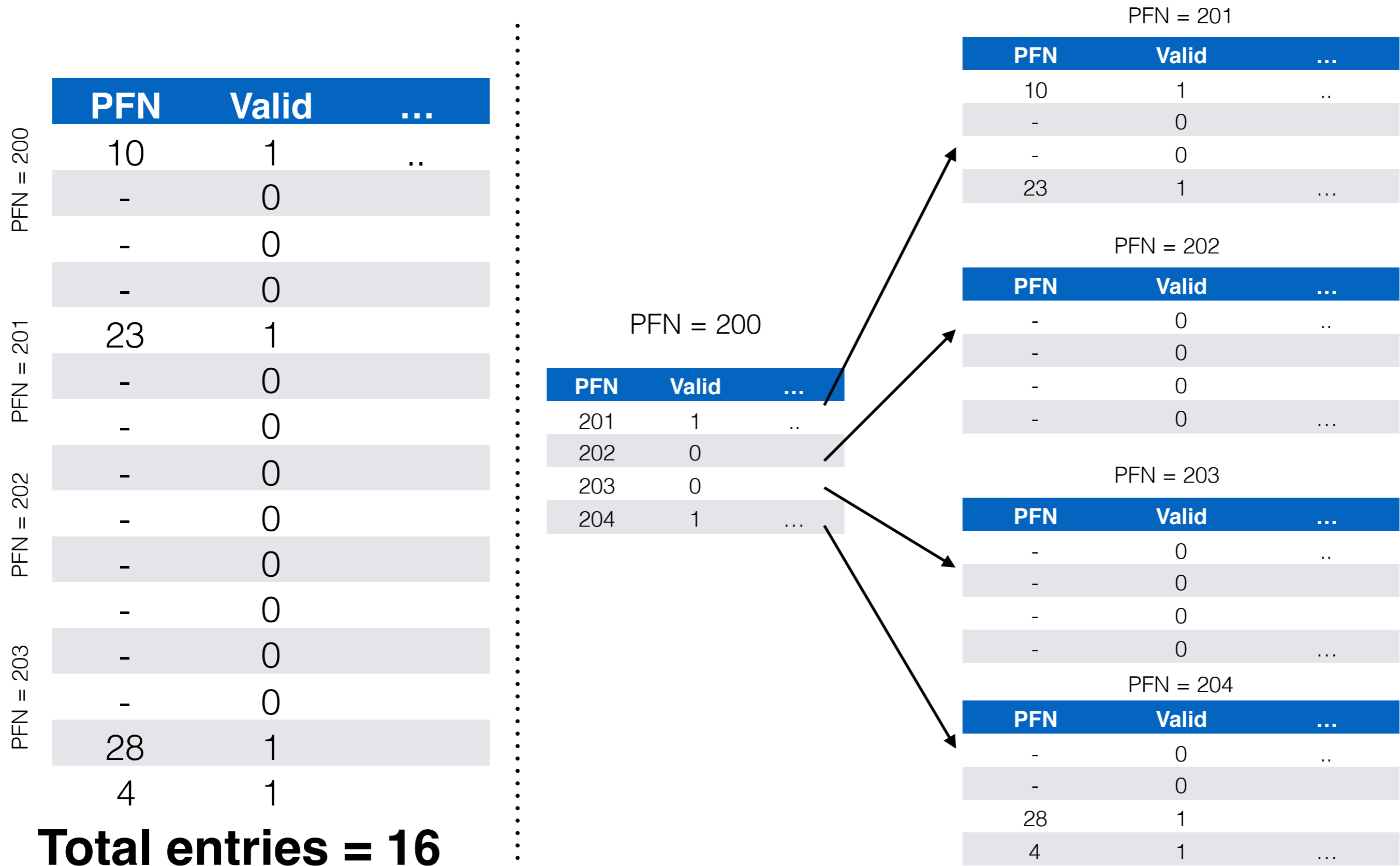
Reducing Memory Overheads of Paging - Segmentation + PT



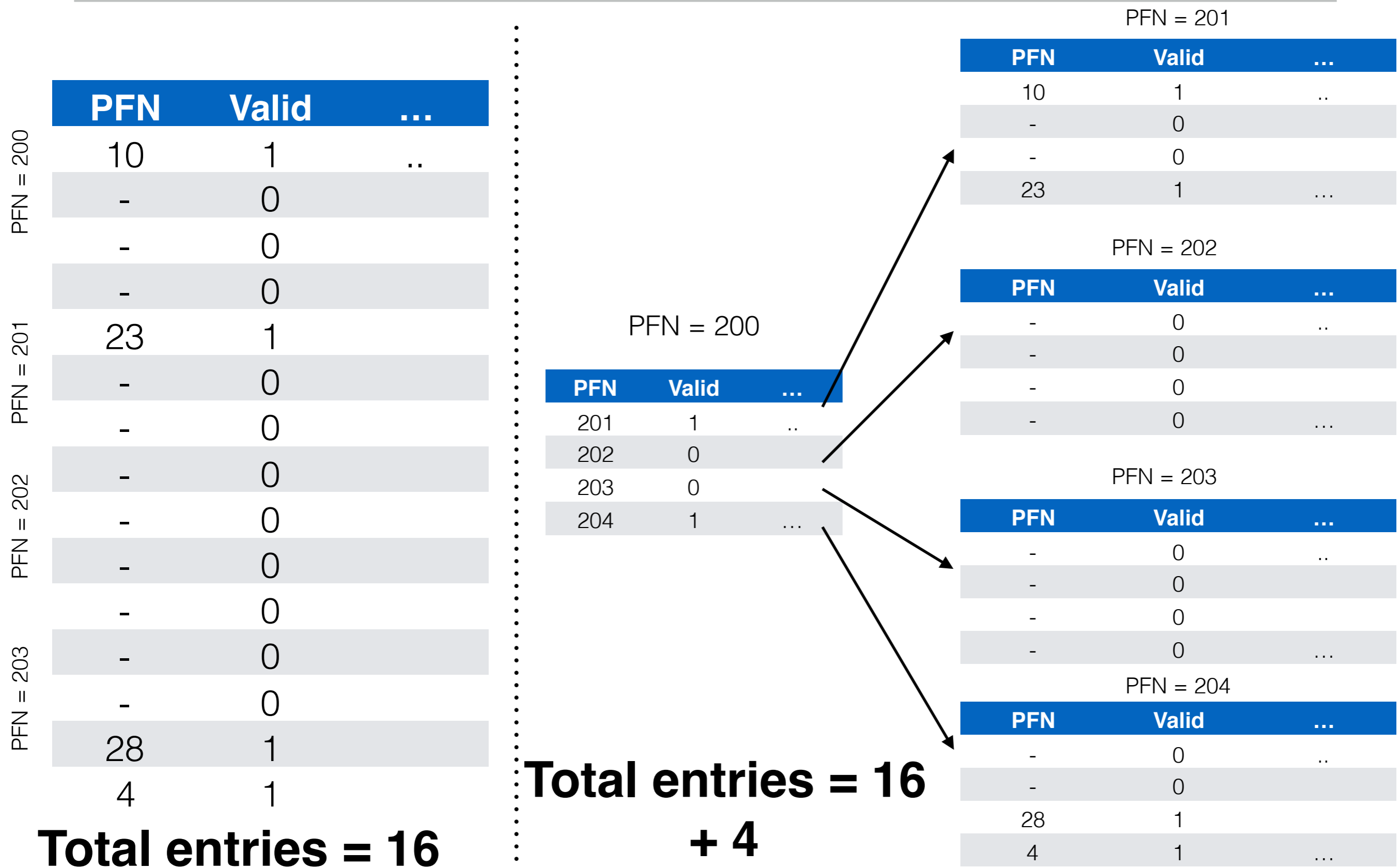
Reducing Memory Overheads of Paging - Segmentation + PT



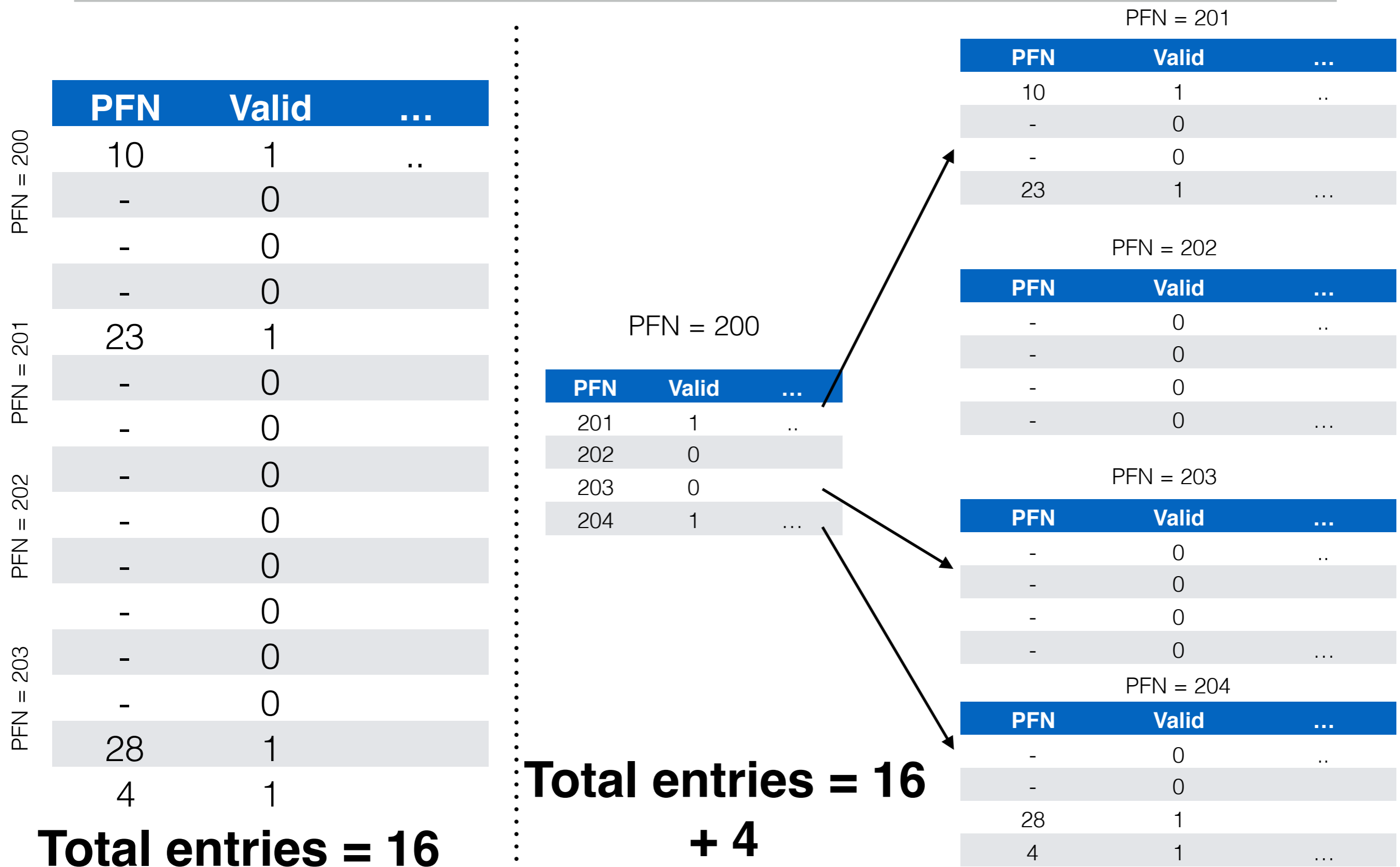
Reducing Memory Overheads of Paging - Segmentation + PT



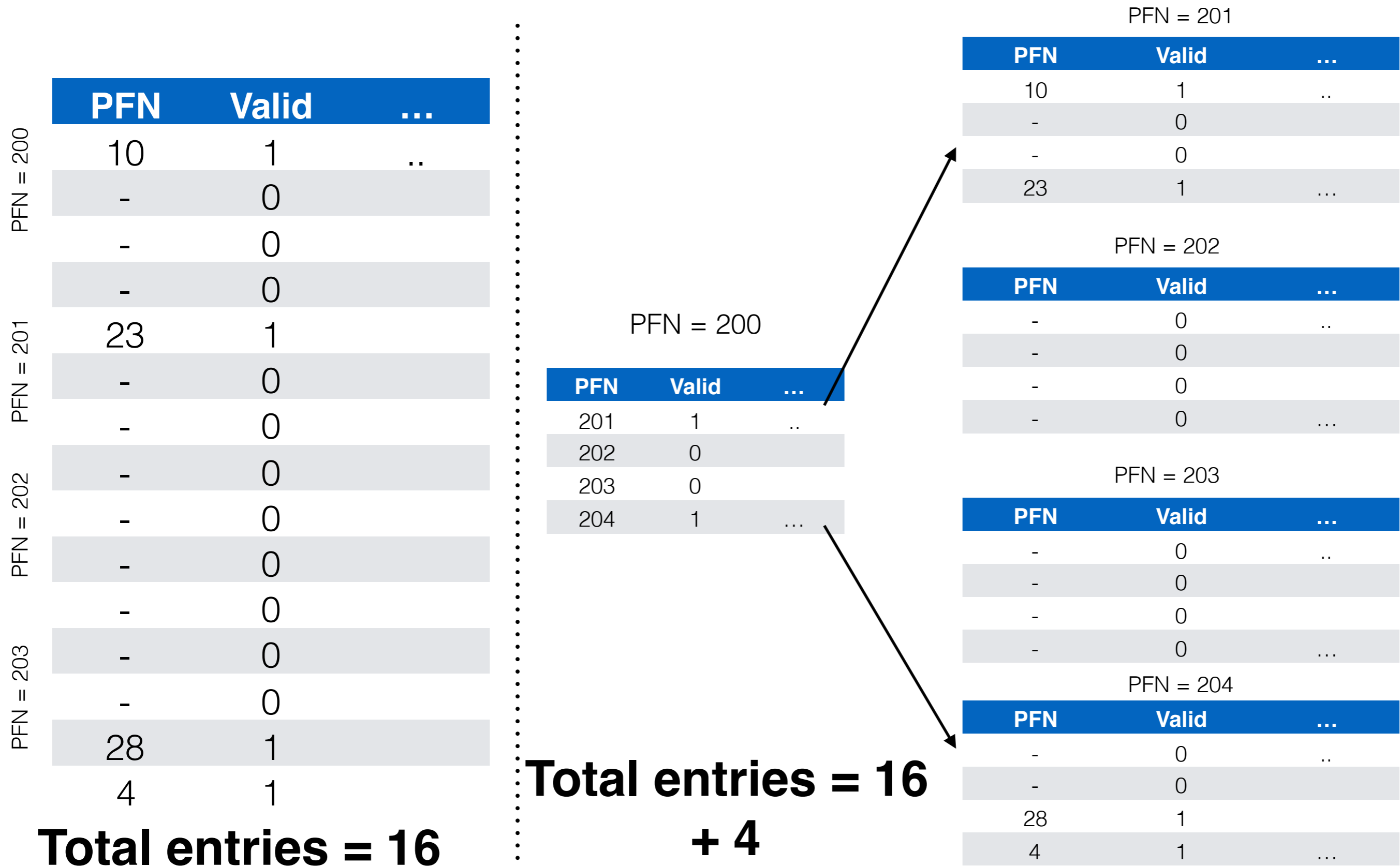
Reducing Memory Overheads of Paging - Segmentation + PT



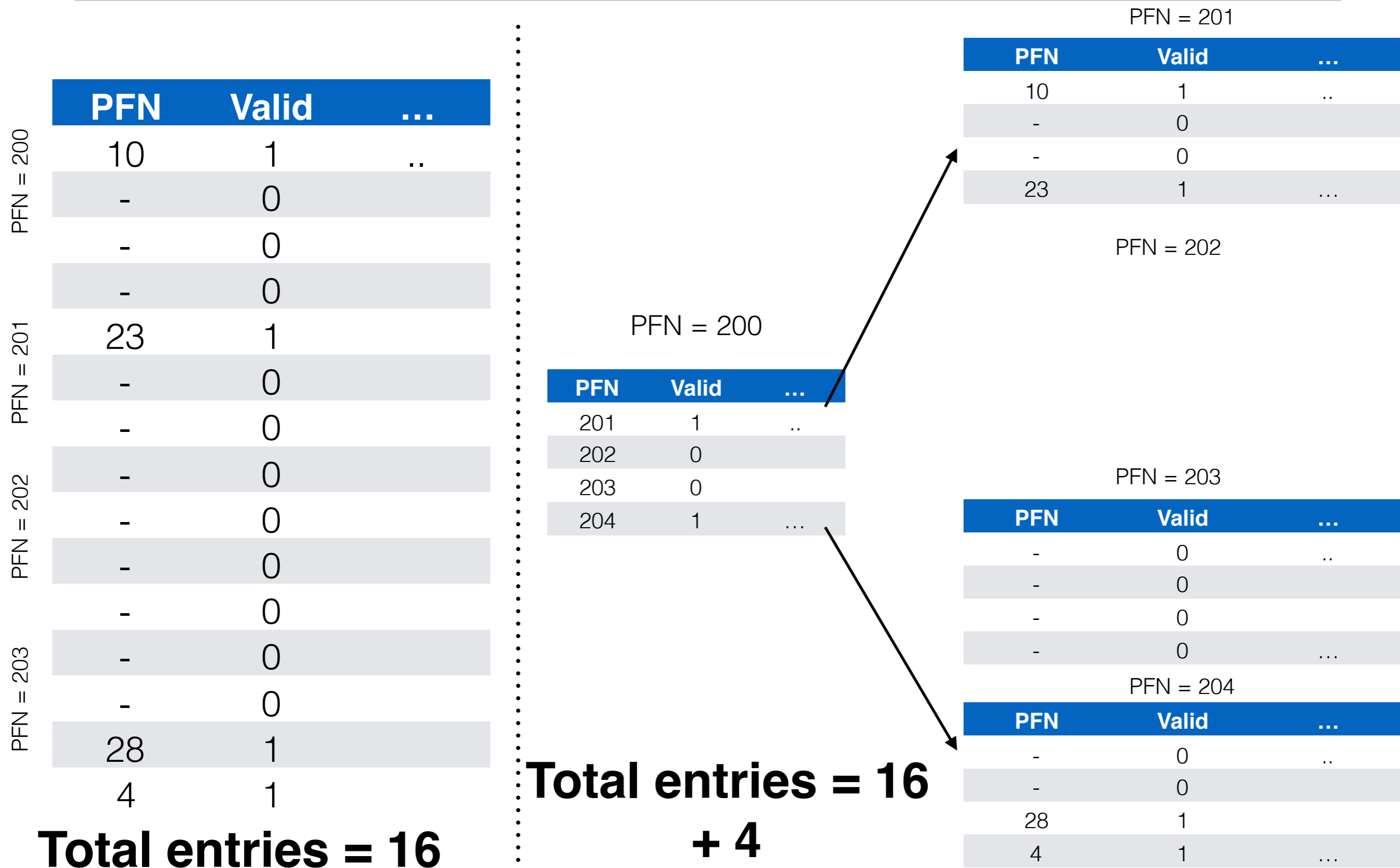
Reducing Memory Overheads of Paging - Segmentation + PT



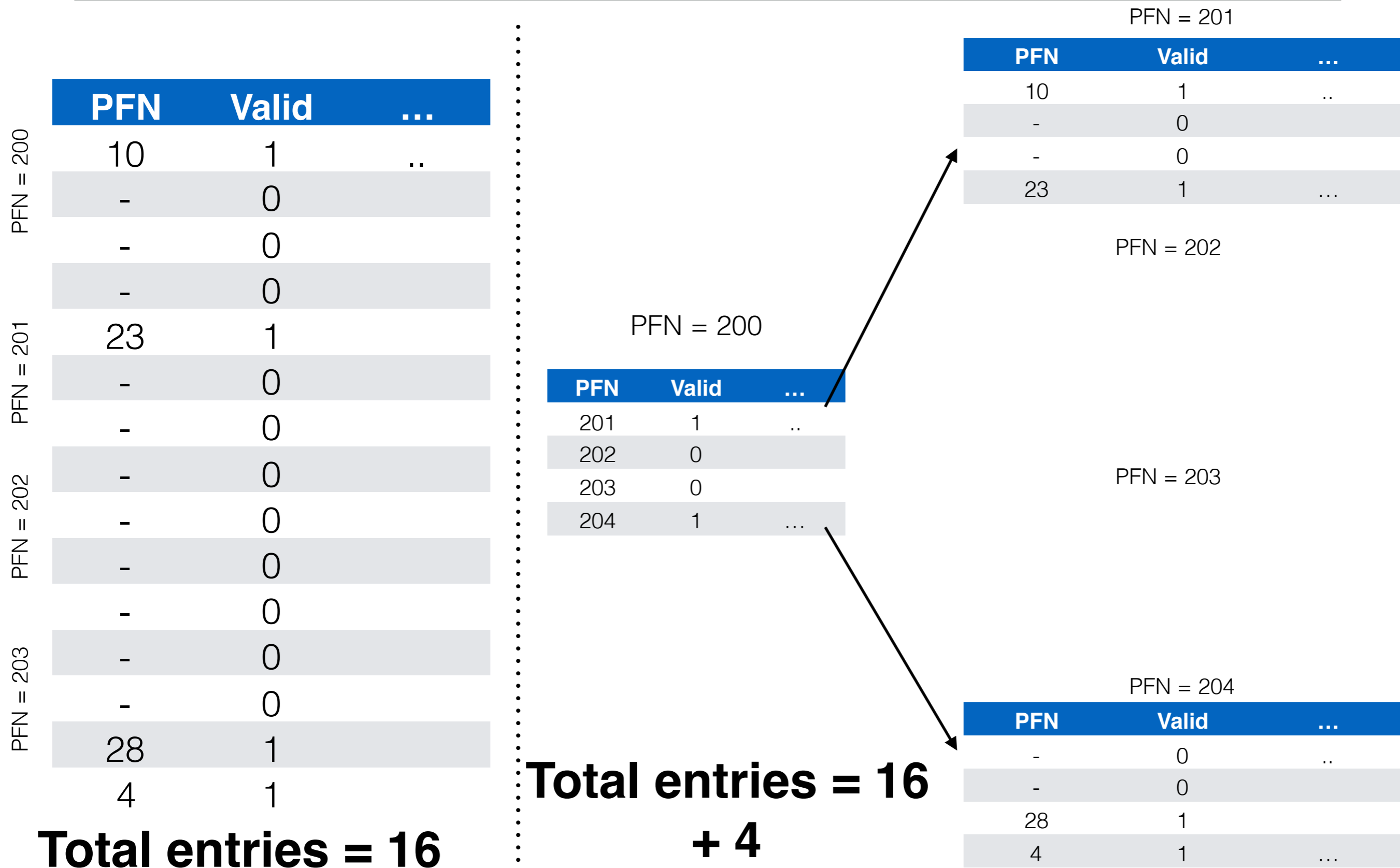
Reducing Memory Overheads of Paging - Segmentation + PT



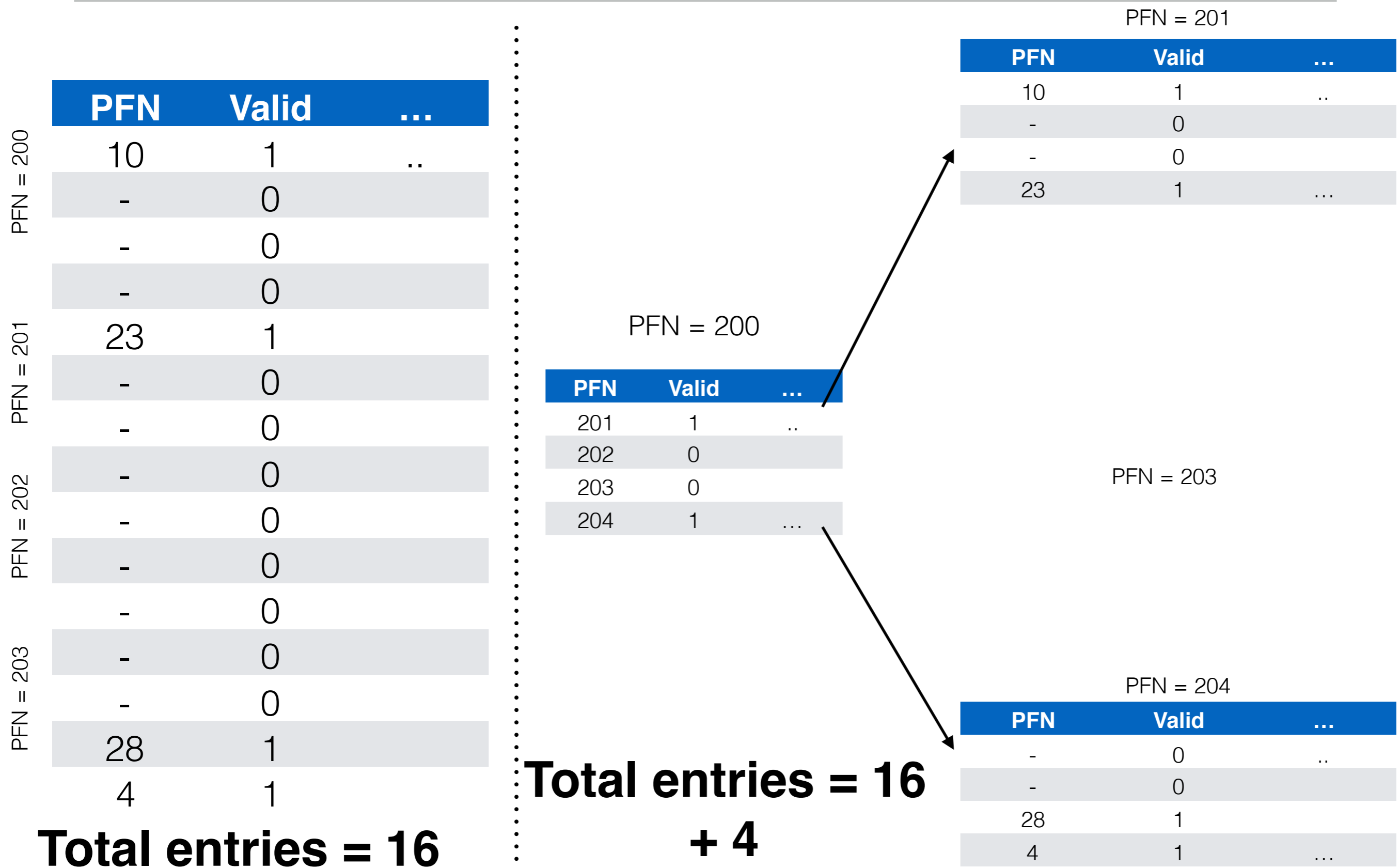
Reducing Memory Overheads of Paging - Segmentation + PT



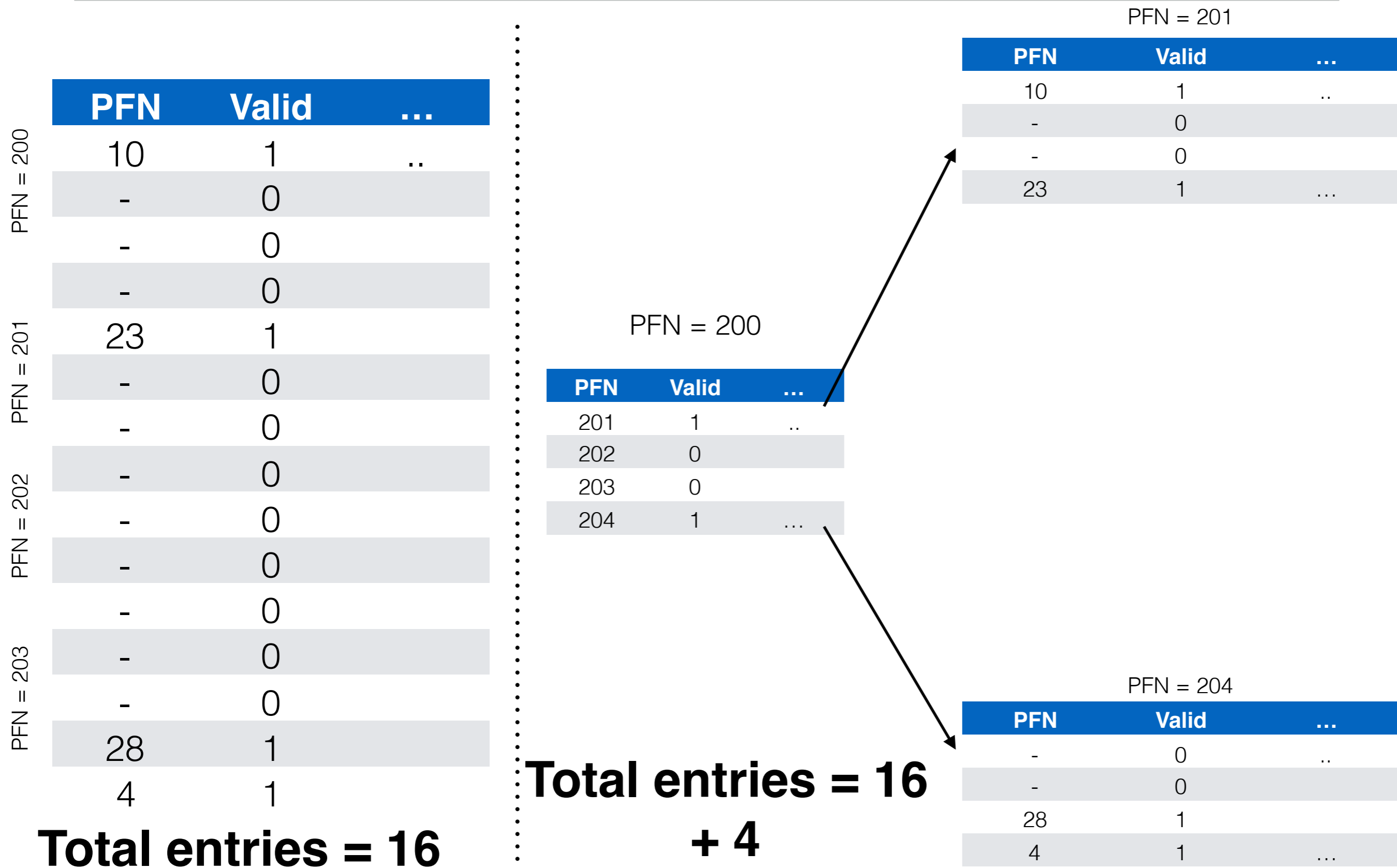
Reducing Memory Overheads of Paging - Segmentation + PT



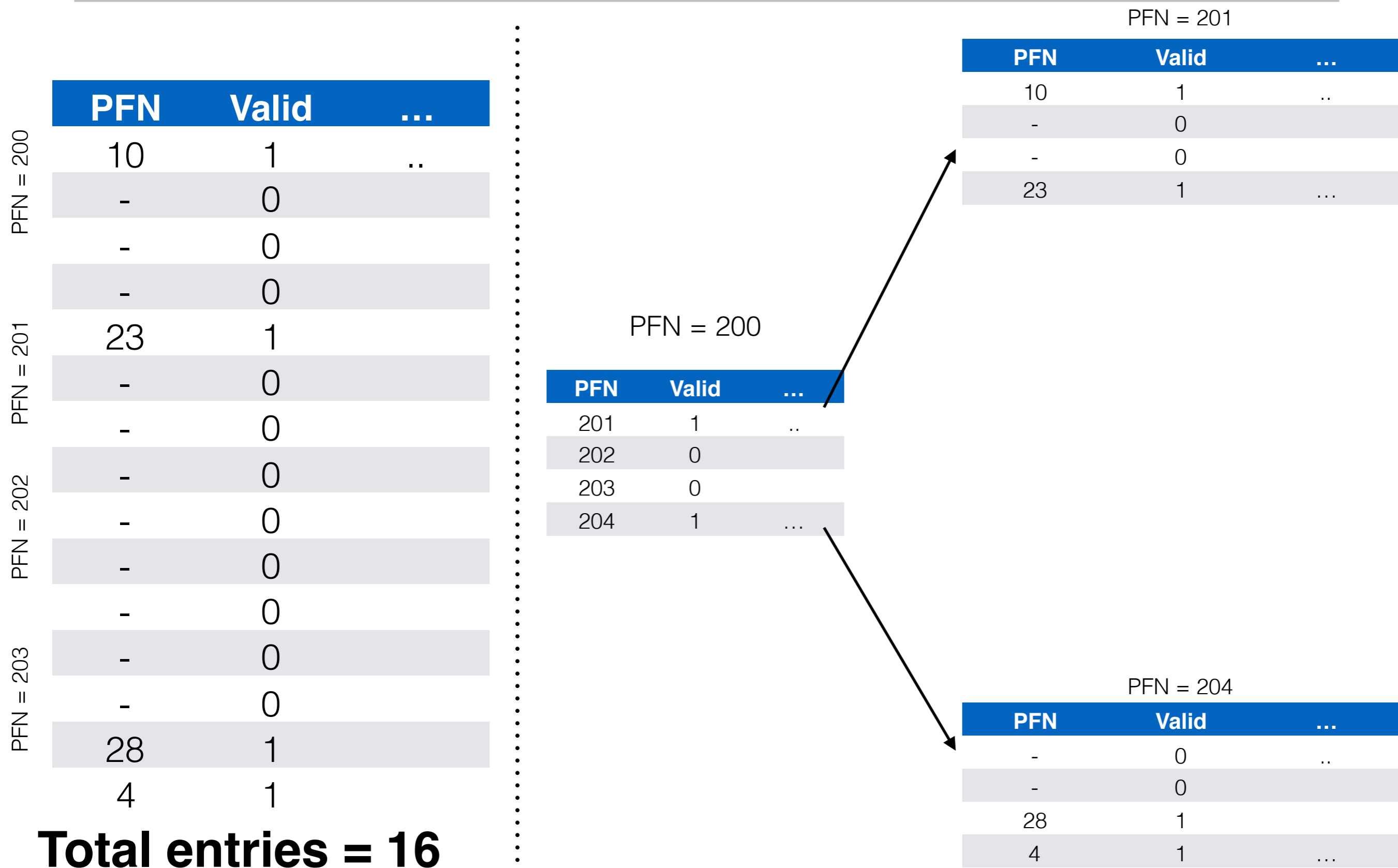
Reducing Memory Overheads of Paging - Segmentation + PT



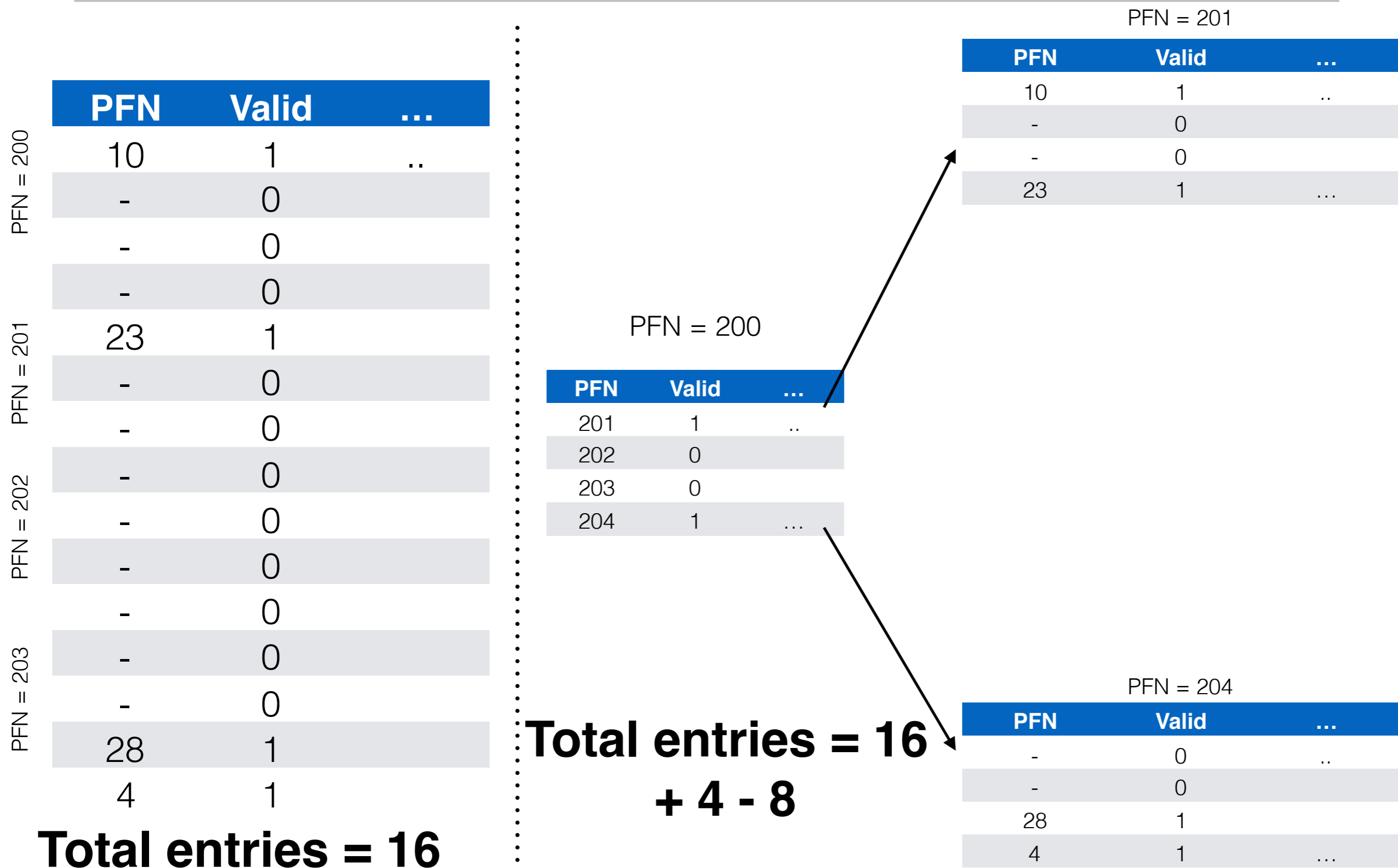
Reducing Memory Overheads of Paging - Segmentation + PT



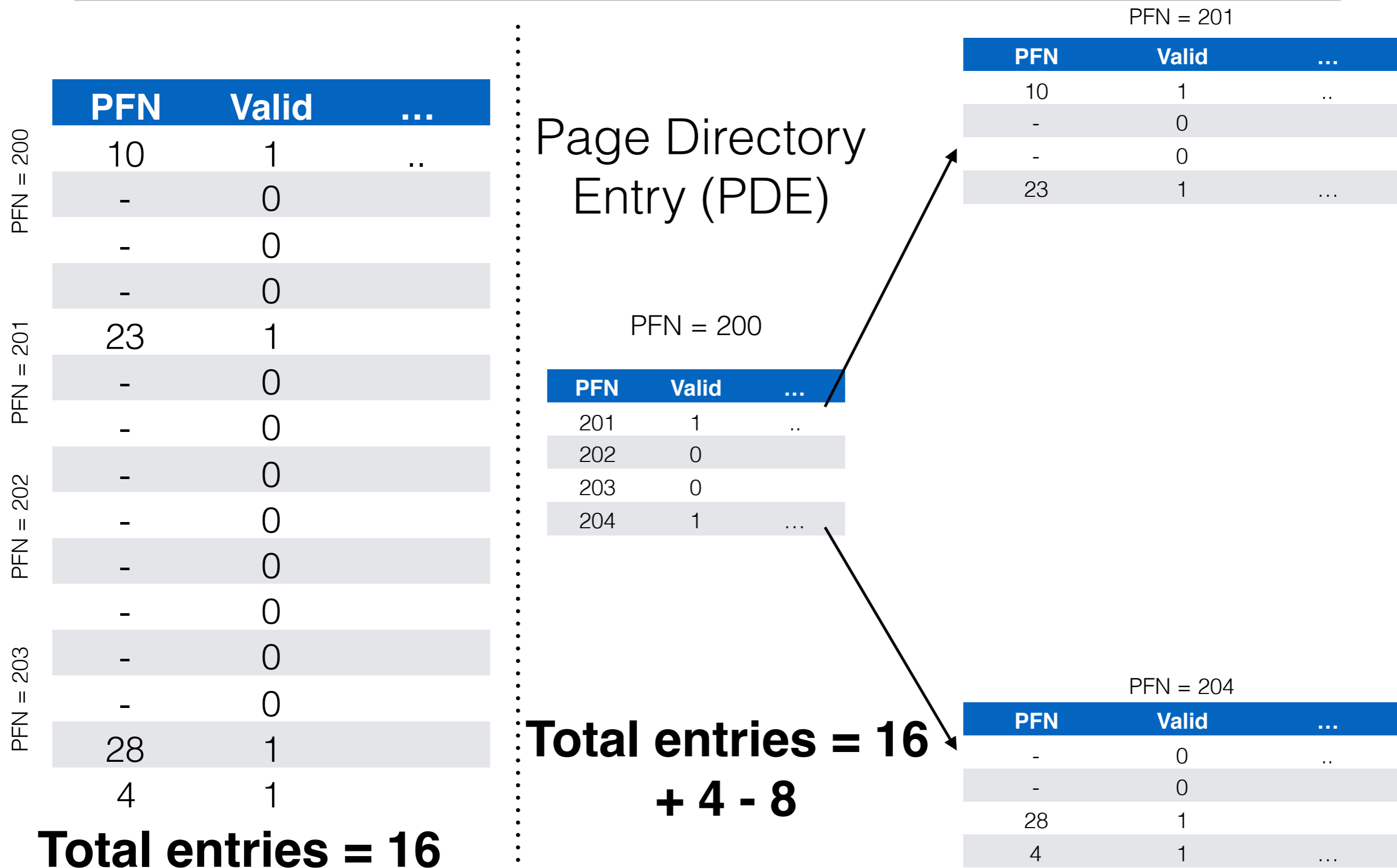
Reducing Memory Overheads of Paging - Segmentation + PT



Reducing Memory Overheads of Paging - Segmentation + PT



Reducing Memory Overheads of Paging - Segmentation + PT



Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging)

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging)

- Multi-level paging allocates memory proportional to requirement

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging)

- Multi-level paging allocates memory proportional to requirement
 - Sparse address spaces well supported

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging)

- Multi-level paging allocates memory proportional to requirement
 - Sparse address spaces well supported
- Previously, we needed 4 MB contiguous space for PT memory

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging)

- Multi-level paging allocates memory proportional to requirement
 - Sparse address spaces well supported
- Previously, we needed 4 MB contiguous space for PT memory
 - Now?

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging)

- Multi-level paging allocates memory proportional to requirement
 - Sparse address spaces well supported
- Previously, we needed 4 MB contiguous space for PT memory
 - Now?
 - Can place Page Tables anywhere in memory with multi-level paging...

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

- Address space = 16 KB

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

- Address space = 16 KB
 - Bits needed = 14 bits

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

- Address space = 16 KB
 - Bits needed = 14 bits
- Page size = 64 bytes (Offset)

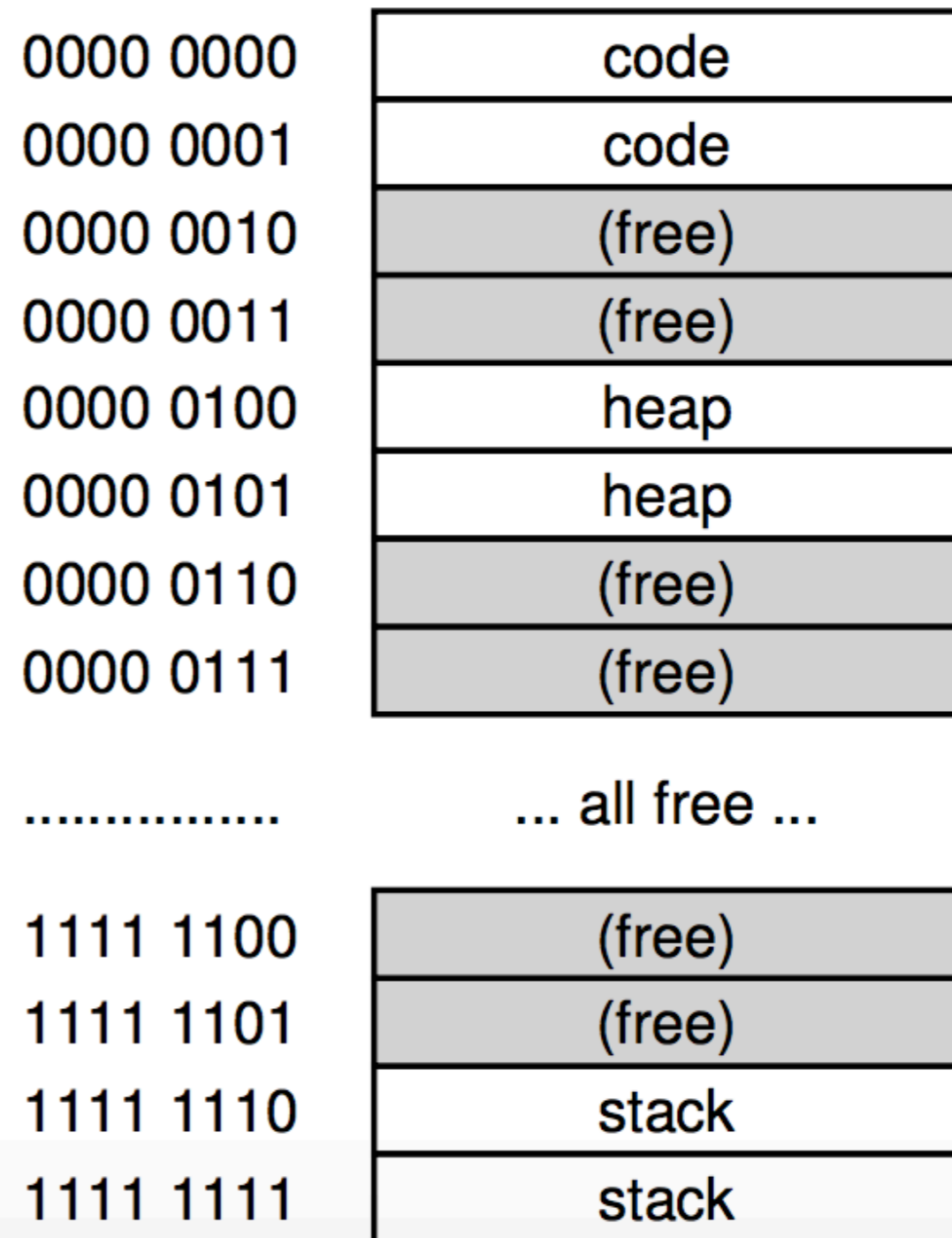
Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

- Address space = 16 KB
 - Bits needed = 14 bits
- Page size = 64 bytes (Offset)
 - 6 bits

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

- Address space = 16 KB
 - Bits needed = 14 bits
- Page size = 64 bytes (Offset)
 - 6 bits
- # Pages = $2^{(14-6)} = 256$ (VPN)

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example



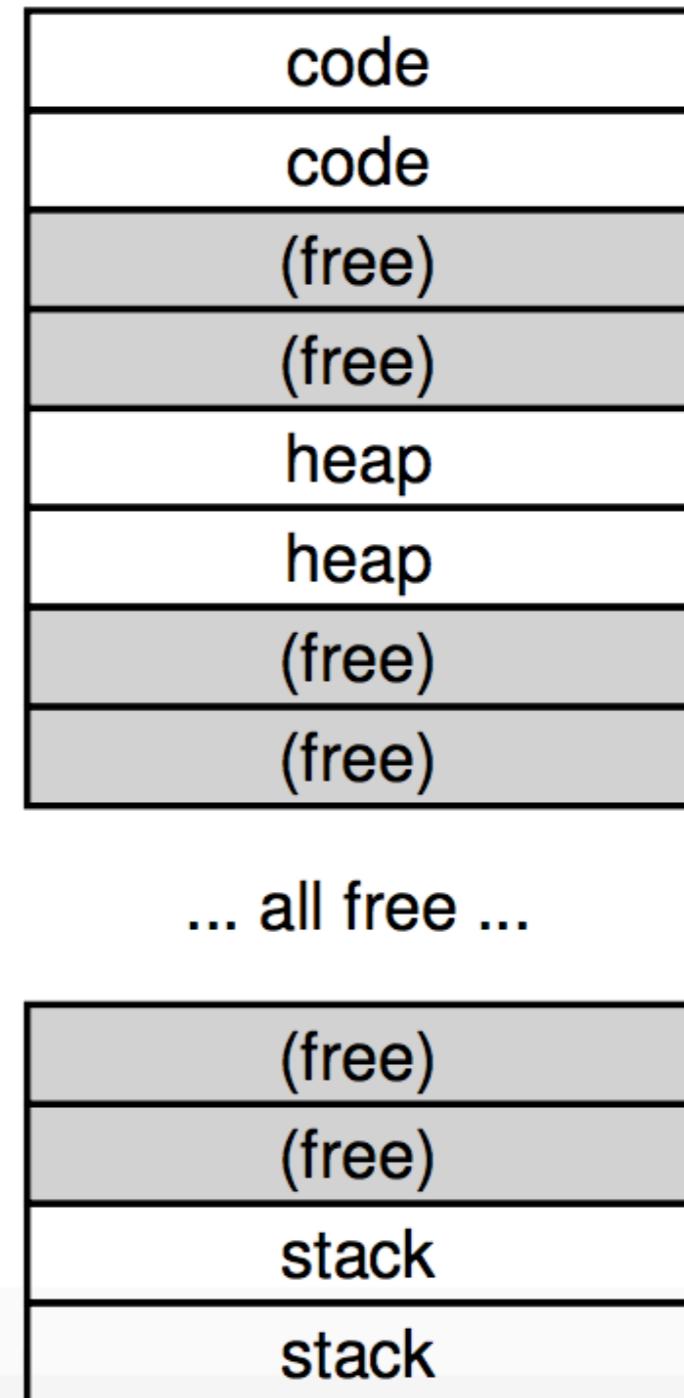
Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Table

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

0000 0000
 0000 0001
 0000 0010
 0000 0011
 0000 0100
 0000 0101
 0000 0110
 0000 0111

 1111 1100
 1111 1101
 1111 1110
 1111 1111



Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Table

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Table

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Table

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	



PTE = 4 bytes

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Table

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	


PTE = 4 bytes

PT size =
 $256 * 4$ bytes
= 1 KB

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Table

PT size =
 $256 * 4$ bytes
= 1 KB



VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

→ PTE = 4 bytes

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Table

PT size =
 $256 * 4$ bytes
= 1 KB

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

PTE = 4 bytes

Page size =
64 bytes =
16 PTE

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Table

PT size =
 $256 * 4$ bytes
= 1 KB

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

PTE = 4 bytes

Page size =
64 bytes =
16 PTE

Total PT fits in
 $256 / 16 = 16$
pages

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Table

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Total PT fits in
 $256/16 = 16$
pages

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Table

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Total PT fits in
 $256/16 = 16$
pages

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Table

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

→ PFN = 100

Total PT fits in
 $256/16 = 16$
pages

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Table

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

→ PFN = 100

Total PT fits in
 $256/16 = 16$
pages

→ PFN = 101

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

Page Table

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

→ PFN = 100

Total PT fits in
 $256/16 = 16$
 pages

→ PFN = 101

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

Page Table

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

→ PFN = 100

Total PT fits in
 $256/16 = 16$
 pages

→ PFN = 101

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

Page Table

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0
PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

Page Table

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0
PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

Page Table

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN = 101

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

Page Table

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
101	1	

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

- Address space = 16 KB

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
101	1	

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

- Address space = 16 KB
 - Bits needed = 14 bits

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
101	1	

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

- Address space = 16 KB
 - Bits needed = 14 bits
- Page size = 64 bytes (Offset)

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
101	1	

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

- Address space = 16 KB
 - Bits needed = 14 bits
- Page size = 64 bytes (Offset)
 - 6 bits

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
101	1	

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
101	1	

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Address space = 16 KB
 - Bits needed = 14 bits
- Page size = 64 bytes (Offset)
 - 6 bits
- # Pages = $2^{(14-6)} = 256$ (VPN)

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
101	1	

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Address space = 16 KB
 - Bits needed = 14 bits
- Page size = 64 bytes (Offset)
 - 6 bits
- # Pages = $2^{(14-6)} = 256$ (VPN)
- How to allocate remaining $(14-6) = 8$ bits?

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

- Number of entries in each page = 16

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
101	1	

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Address space = 16 KB
 - Bits needed = 14 bits
- Page size = 64 bytes (Offset)
 - 6 bits
- # Pages = $2^{(14-6)} = 256$ (VPN)
- How to allocate remaining $(14-6) = 8$ bits?

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

- Number of entries in each page = 16
- Number of bits required = 4

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
101	1	

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Address space = 16 KB
 - Bits needed = 14 bits
- Page size = 64 bytes (Offset)
 - 6 bits
- # Pages = $2^{(14-6)} = 256$ (VPN)
- How to allocate remaining $(14-6) = 8$ bits?

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

- Number of entries in PDE = 16

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
101	1	

- Number of entries in each page = 16
- Number of bits required = 4

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Address space = 16 KB
 - Bits needed = 14 bits
- Page size = 64 bytes (Offset)
 - 6 bits
- # Pages = $2^{(14-6)} = 256$ (VPN)
- How to allocate remaining $(14-6) = 8$ bits?

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
	-	0	
	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

- Number of entries in PDE = 16
- Number of bits required = 4

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
101	1	

- Number of entries in each page = 16
- Number of bits required = 4

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

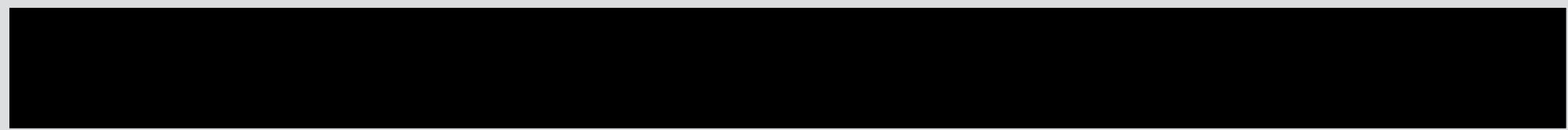
PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Address space = 16 KB
 - Bits needed = 14 bits
- Page size = 64 bytes (Offset)
 - 6 bits
- # Pages = $2^{(14-6)} = 256$ (VPN)
- How to allocate remaining $(14-6) = 8$ bits?

VPN	PFN	Valid	...
0	10	1	..
1	23	1	
-	-	0	
-	-	0	
4	80	1	
5	59	1	
...	..	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
...	...	0	
254	55	1	
255	45	1	

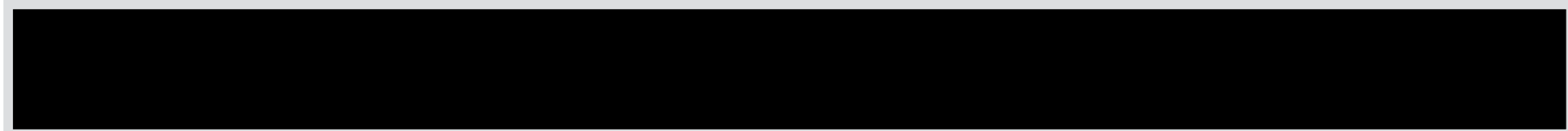
Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

14 bit VA space



Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

14 bit VA space



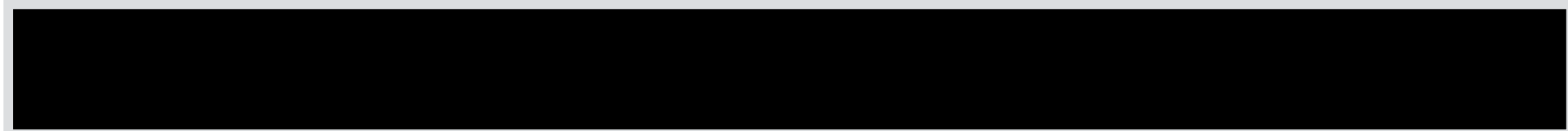
8 bit VPN

6 bit Offset



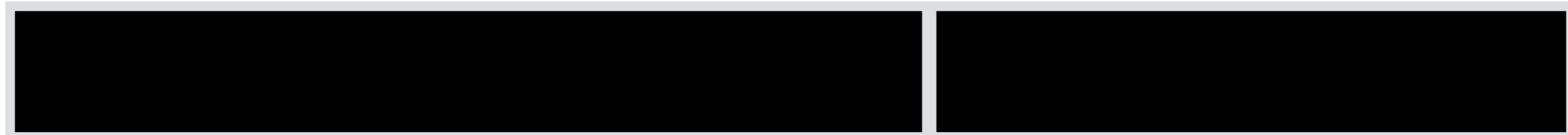
Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

14 bit VA space



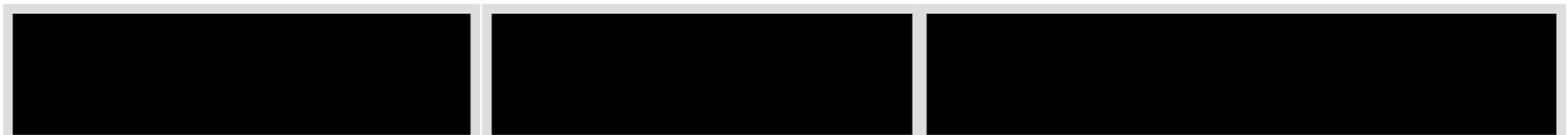
8 bit VPN

6 bit Offset

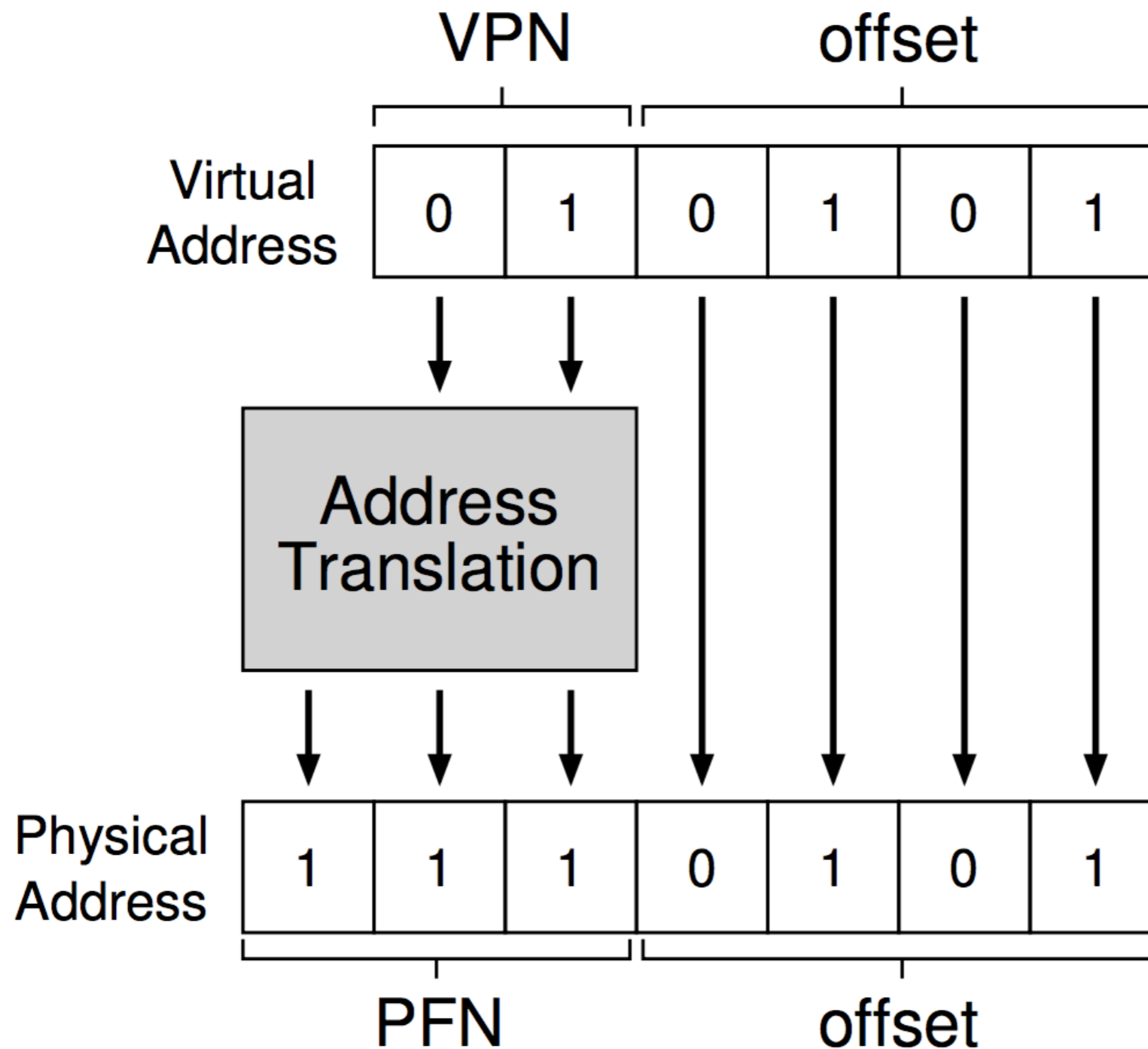


4 bit PD Index 4 bit PT Index

6 bit Offset



Address Translation Summary



Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN = 101

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN = 101

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Translate 0x3F80 VA to PA

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN = 101

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Translate 0x3F80 VA to PA
- $0x3F80 = (11)(1111)(1000)(0000)$

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN = 101

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Translate 0x3F80 VA to PA
- 0x3F80 = (11)(1111)(1000)(0000)
- **(11)(1111)(1000)(0000)** - >PD Index

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN = 101

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Translate 0x3F80 VA to PA
- $0x3F80 = (11)(1111)(1000)(0000)$
- **(11)(1111)(1000)(0000)** -> PD Index
- PD index = 1111 -> PFN = 101

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN = 101

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Translate 0x3F80 VA to PA
- $0x3F80 = (11)(1111)(1000)(0000)$
- **$(11)(1111)(1000)(0000)$** -> PD Index
- PD index = 1111 -> PFN = 101

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN = 101

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Translate 0x3F80 VA to PA
- $0x3F80 = (11)(1111)(1000)(0000)$
- **(11)(1111)(1000)(0000)** -> PD Index
- PD index = 1111 -> PFN = 101

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN = 101

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Translate 0x3F80 VA to PA
- $0x3F80 = (11)(1111)(1000)(0000)$
- **$(11)(1111)(1000)(0000)$** - > PD Index
- PD index = 1111 -> PFN = 101
- Next 4 bits = PT index = 1110 = 14 decimal -> last PTE in PFN = 101 -> PFN = 55

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN = 101

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Translate 0x3F80 VA to PA
- $0x3F80 = (11)(1111)(1000)(0000)$
- **$(11)(1111)(1000)(0000)$** -> PD Index
- PD index = 1111 -> PFN = 101
- Next 4 bits = PT index = 1110 = 14 decimal -> last PTE in PFN = 101 -> PFN = 55
- Offset = last 6 digits = 0

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN = 101

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Translate 0x3F80 VA to PA
- $0x3F80 = (11)(1111)(1000)(0000)$
- **$(11)(1111)(1000)(0000)$** -> PD Index
- PD index = 1111 -> PFN = 101
- Next 4 bits = PT index = 1110 = 14 decimal -> last PTE in PFN = 101 -> PFN = 55
- Offset = last 6 digits = 0
- $PA = 55 \ll \text{SHIFT} + \text{OFFSET} = 55 \ll 6 + 000000 = 0x0DC0$

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN = 101

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Translate 0x3F80 VA to PA
- $0x3F80 = (11)(1111)(1000)(0000)$
- **$(11)(1111)(1000)(0000)$** -> PD Index
- PD index = 1111 -> PFN = 101
- Next 4 bits = PT index = 1110 = 14 decimal -> last PTE in PFN = 101 -> PFN = 55
- Offset = last 6 digits = 0
- $PA = 55 \ll \text{SHIFT} + \text{OFFSET} = 55 \ll 6 + 000000 = 0x0DC0$

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN = 101

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Translate 0x3F80 VA to PA
- $0x3F80 = (11)(1111)(1000)(0000)$
- **$(11)(1111)(1000)(0000)$** -> PD Index
- PD index = 1111 -> PFN = 101
- Next 4 bits = PT index = 1110 = 14 decimal -> last PTE in PFN = 101 -> PFN = 55
- Offset = last 6 digits = 0
- $PA = 55 \ll \text{SHIFT} + \text{OFFSET} = 55 \ll 6 + 000000 = 0x0DC0$

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

PFN = 101

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Translate 0x3F80 VA to PA
- $0x3F80 = (11)(1111)(1000)(0000)$
- **(11)(1111)(1000)(0000)** -> PD Index
- PD index = 1111 -> PFN = 101
- Next 4 bits = PT index = 1110 = 14 decimal -> last PTE in PFN = 101 -> PFN = 55
- Offset = last 6 digits = 0
- $PA = 55 \ll \text{SHIFT} + \text{OFFSET} = 55 \ll 6 + 000000 = 0x0DC0$

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example

Page Directory

PFN	Valid	...
100	1	..
-	0	
-	0	
-	0	
-	0	
-	0	
..	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
...	0	
-	0	
101	1	

PFN = 100

PFN	Valid
10	1
23	1
-	0
-	0
80	1
59	1
..	0
...	0
...	0

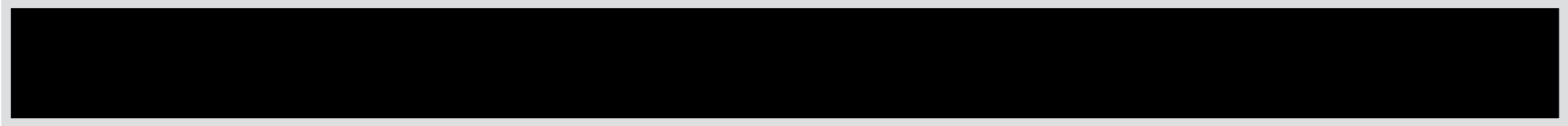
PFN = 101

PFN	Valid
-	0
-	0
-	0
-	0
-	0
-	0
-	0
..	0
55	1
45	1

- Translate 0x3F80 VA to PA
- $0x3F80 = (11)(1111)(1000)(0000)$
- **$(11)(1111)(1000)(0000)$** -> PD Index
- PD index = 1111 -> PFN = 101
- Next 4 bits = PT index = 1110 = 14 decimal -> last PTE in PFN = 101 -> PFN = 55
- Offset = last 6 digits = 0
- $PA = 55 \ll \text{SHIFT} + \text{OFFSET} = 55 \ll 6 + 000000 = 0x0DC0$

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example 2

30 bit VA space



Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example 2

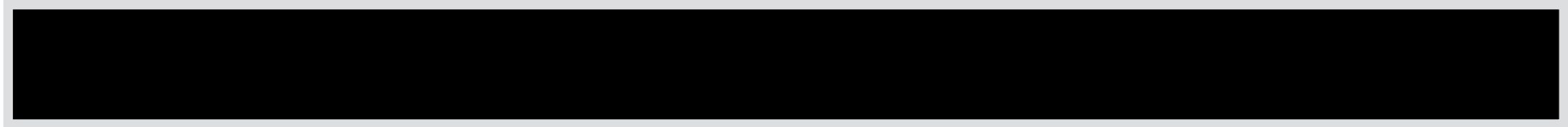
30 bit VA space



512 byte page, 4 byte PTE

Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example 2

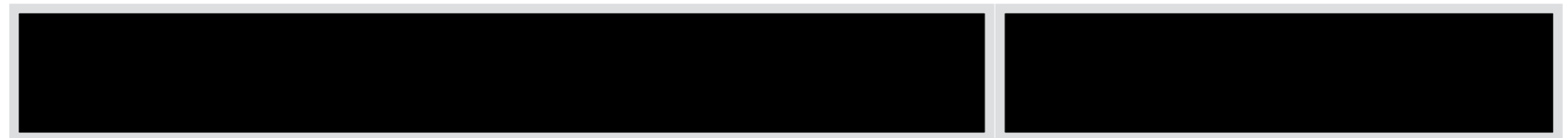
30 bit VA space



512 byte page, 4 byte PTE

21 bit VPN

9 bit Offset



Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example 2

30 bit VA space



512 byte page, 4 byte PTE

21 bit VPN

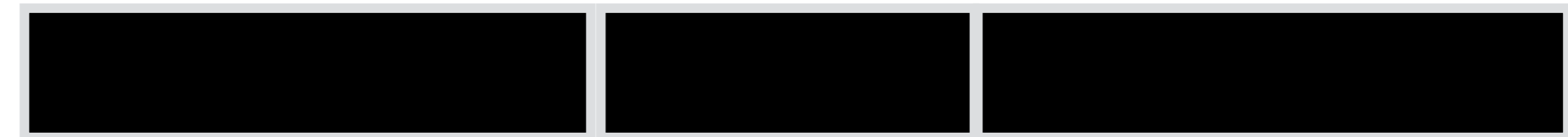
9 bit Offset



14 bit PD Index

7 bit PT Index

9 bit Offset



Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example 2

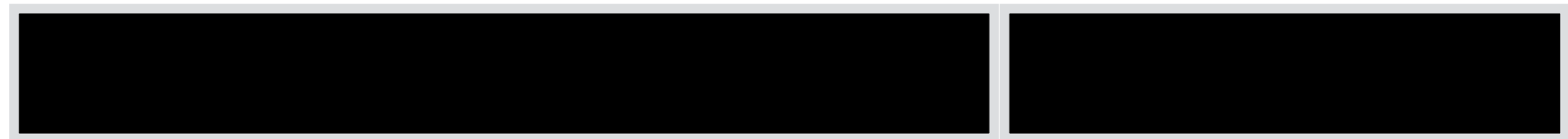
30 bit VA space



512 byte page, 4 byte PTE

21 bit VPN

9 bit Offset

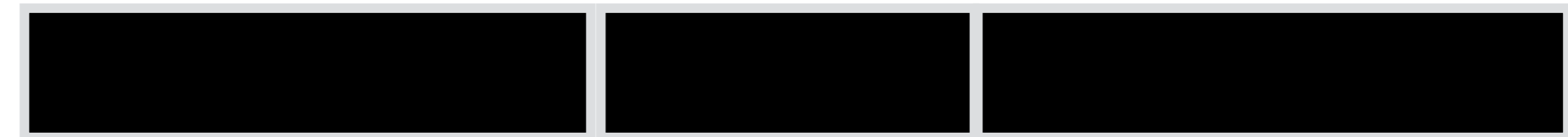


128 PTE per page

14 bit PD Index

7 bit PT Index

9 bit Offset



Reducing Memory Overheads of Paging - PT + PT (Multi-level Paging) Example 2

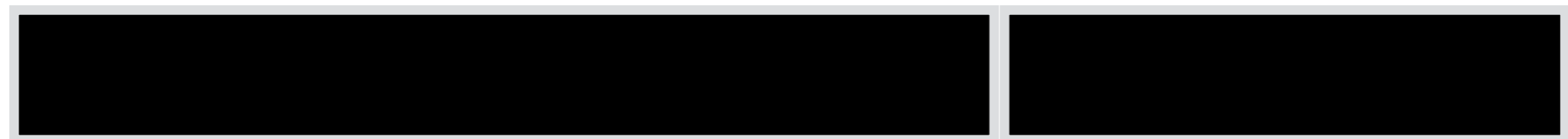
30 bit VA space



512 byte page, 4 byte PTE

21 bit VPN

9 bit Offset

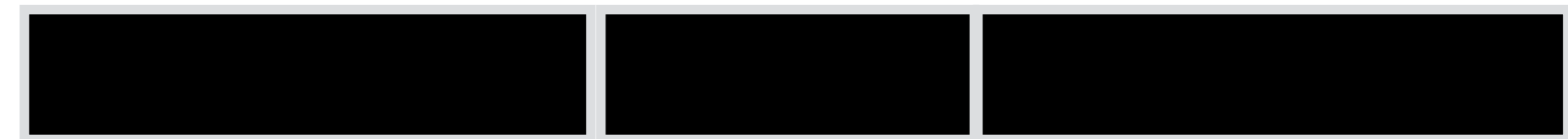


128 PTE per page

14 bit PD Index

7 bit PT Index

9 bit Offset

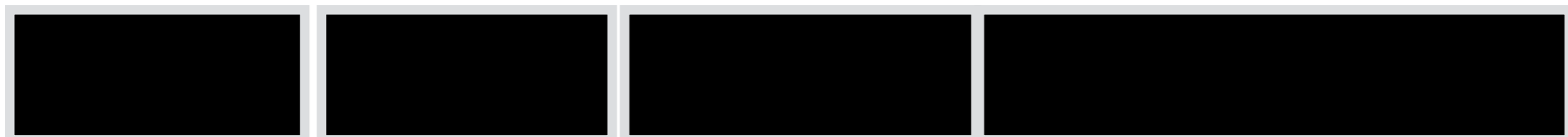


7 bit PD2
Index

7 bit PD1
Index

7 bit PT Index

9 bit Offset



Beyond Physical Memory

scikit-learn.org/stable/modules/scaling_strategies.html

Google Analytics Outlook Web App imac - Google Sear...



Home Installation Documentation Examples

Google Custom Search



Previous 5.6.8. RCv1 dataset Next 7. Computatio.. Up User Guide

scikit-learn v0.19.2
Other versions

Please **cite us** if you use
the software.

6. Strategies to scale
computationally: bigger data
6.1. Scaling with instances using
out-of-core learning

- 6.1.1. Streaming instances
- 6.1.2. Extracting features
- 6.1.3. Incremental learning
- 6.1.4. Examples
- 6.1.5. Notes

6. Strategies to scale computationally: bigger data

For some applications the amount of examples, features (or both) and/or the speed at which they need to be processed are challenging for traditional approaches. In these cases scikit-learn has a number of options you can consider to make your system scale.

6.1. Scaling with instances using out-of-core learning

Out-of-core (or “external memory”) learning is a technique used to learn from data that cannot fit in a computer’s main memory (RAM).

Here is sketch of a system designed to achieve this goal:

1. a way to stream instances
2. a way to extract features from instances
3. an incremental algorithm

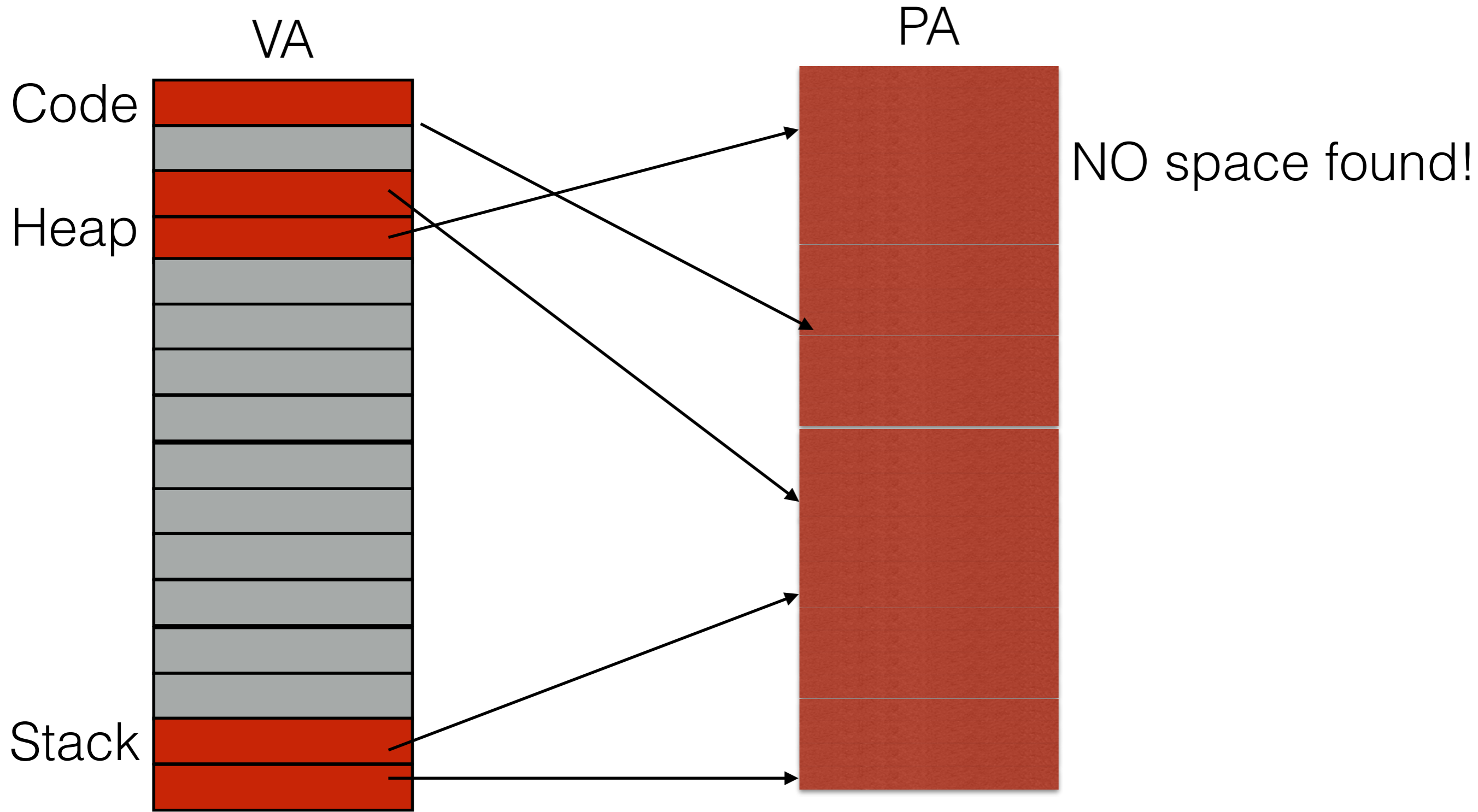
6.1.1. Streaming instances

Basically, 1. may be a reader that yields instances from files on a hard drive, a database, from a network stream etc. However, details on how to achieve this are beyond the scope of this documentation.

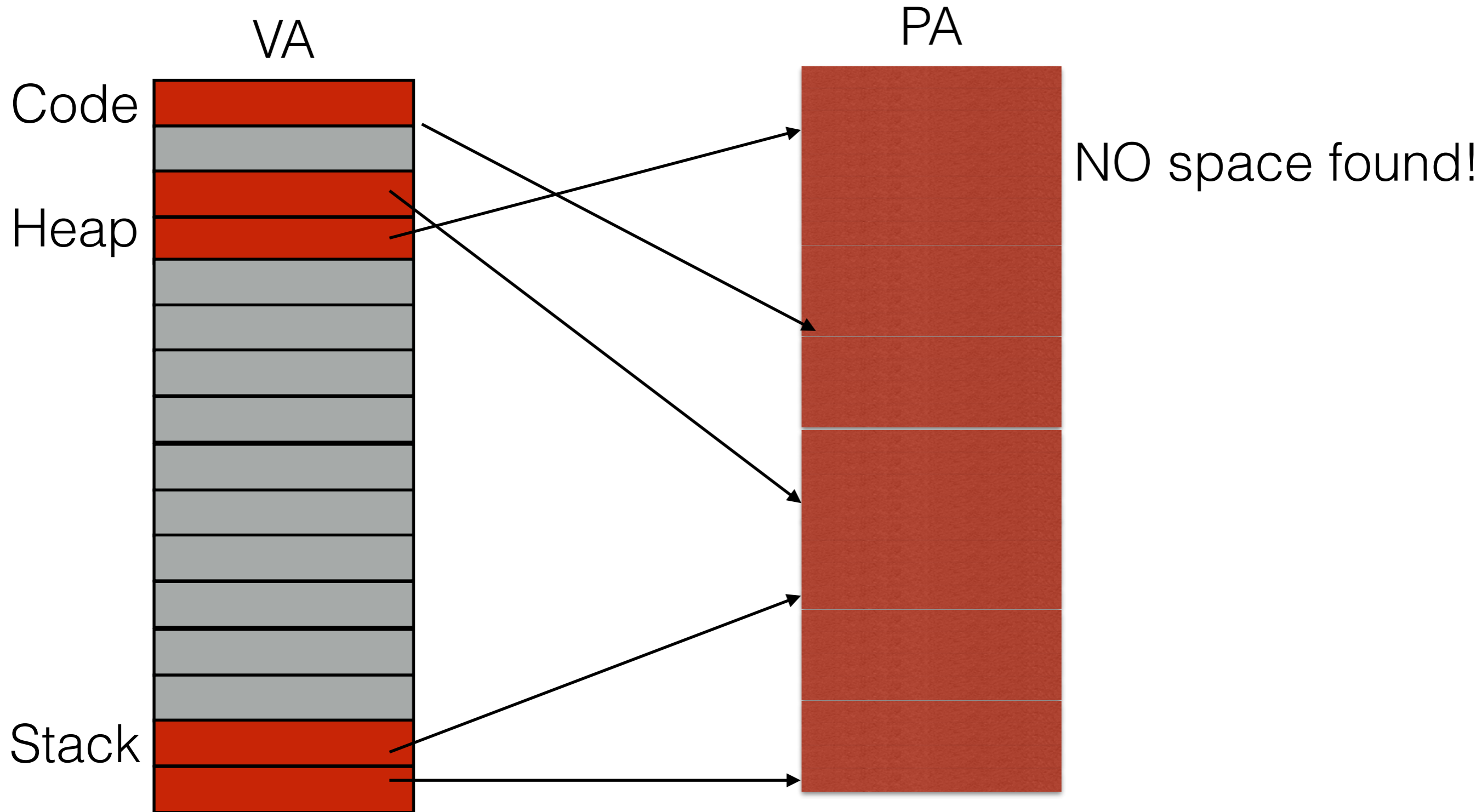
6.1.2. Extracting features

2. could be any relevant way to extract features among the different [feature extraction](#) methods supported by scikit-learn. However, when working with data that needs vectorization and where the set of features or values is not known in advance one should take explicit care. A good example is text classification where unknown terms are likely to be found during training. It is possible to use a stateful vectorizer if making multiple passes over the data is reasonable from an application point of view. Otherwise, one can turn up the difficulty by using a stateless feature extractor. Currently the preferred way to do this

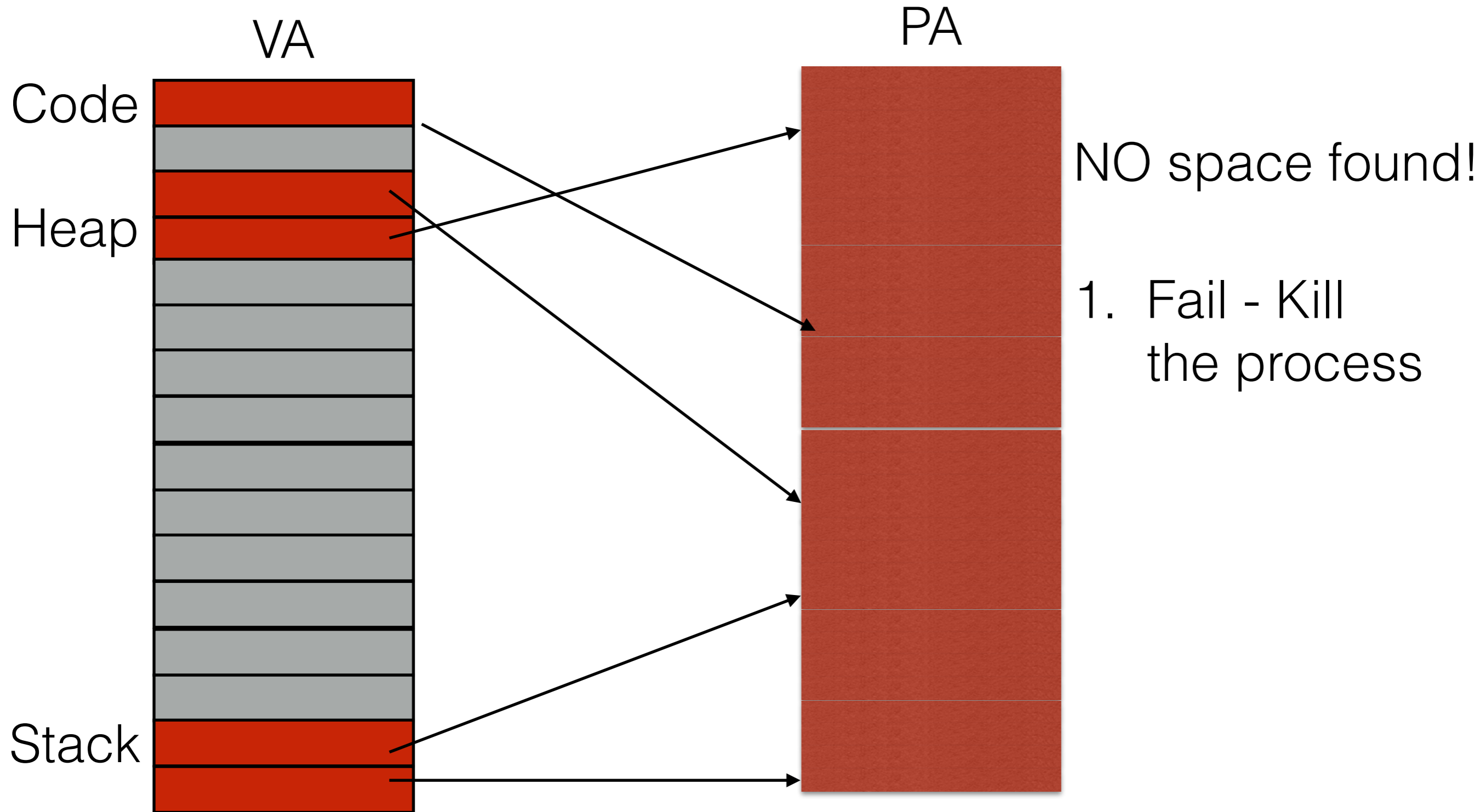
Out of PA space



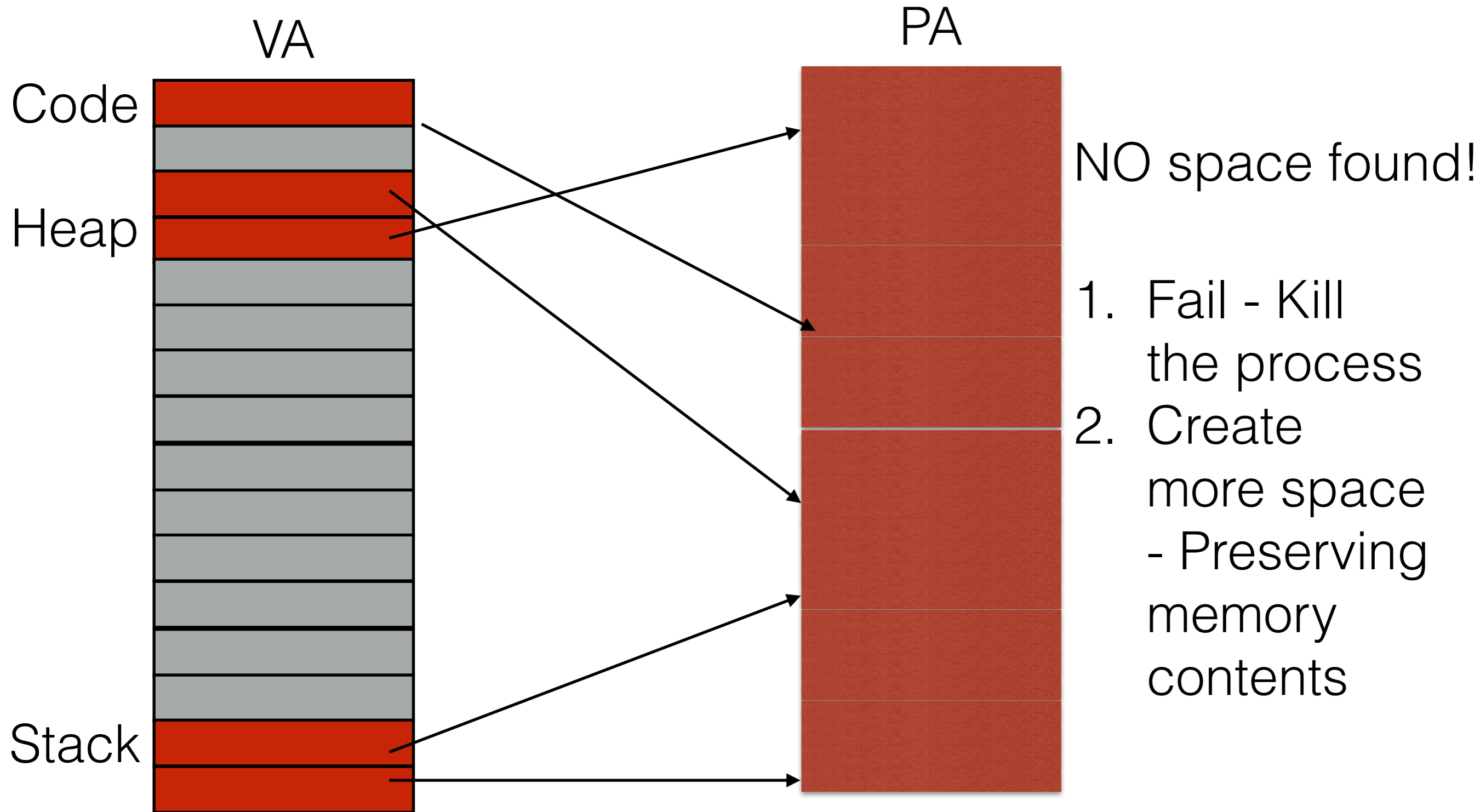
Out of PA space



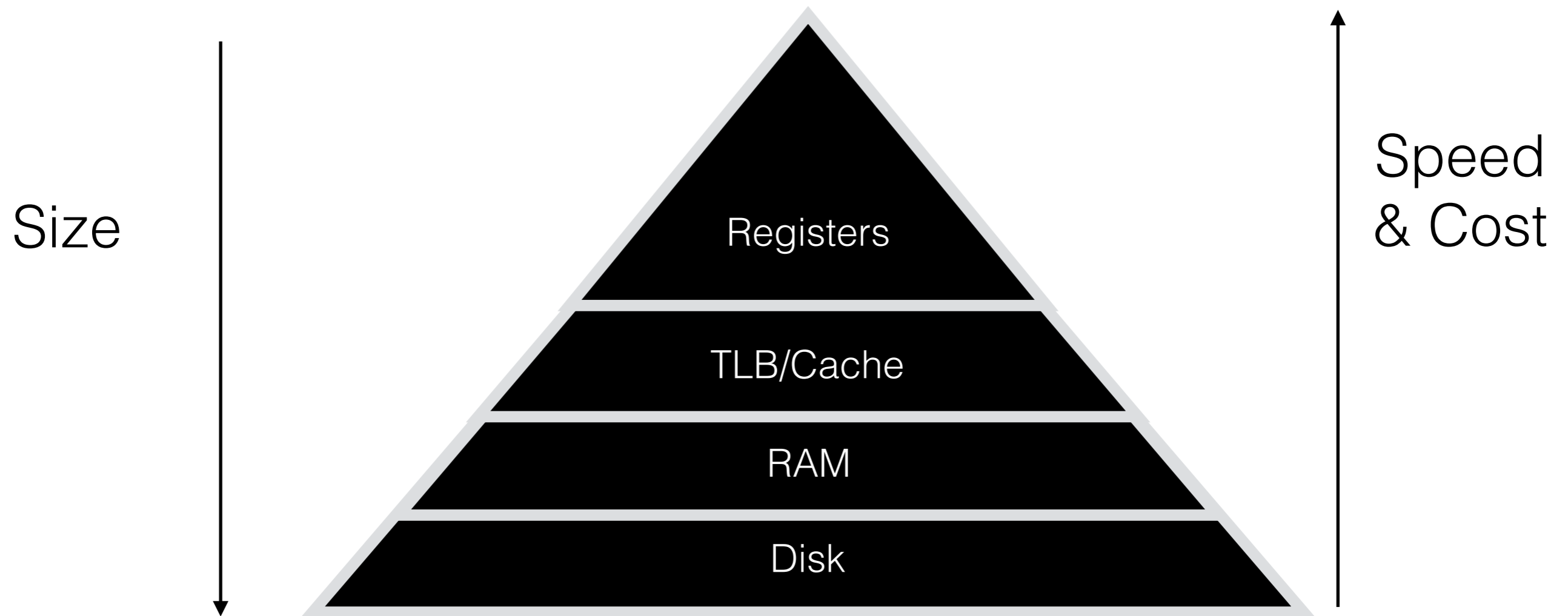
Out of PA space



Out of PA space



Out of PA space - Memory hierarchy



Out of PA space - Virtualisation

Out of PA space - Virtualisation

1. Does the user know whether the PFN came from TLB or memory?

Out of PA space - Virtualisation

1. Does the user know whether the PFN came from TLB or memory?
 - No!

Out of PA space - Virtualisation

1. Does the user know whether the PFN came from TLB or memory?
 - No!
 - Does the user even know if paging is used?

Out of PA space - Virtualisation

1. Does the user know whether the PFN came from TLB or memory?
 - No!
 - Does the user even know if paging is used?
2. Should the user know if the page came from memory or TLB or disk?

Out of PA space - Virtualisation

1. Does the user know whether the PFN came from TLB or memory?
 - No!
 - Does the user even know if paging is used?
2. Should the user know if the page came from memory or TLB or disk?
 - No!

Out of PA space - Virtualisation

1. Does the user know whether the PFN came from TLB or memory?
 - No!
 - Does the user even know if paging is used?
2. Should the user know if the page came from memory or TLB or disk?
 - No!
 - The last time the process used the virtual address, it behaved like memory.

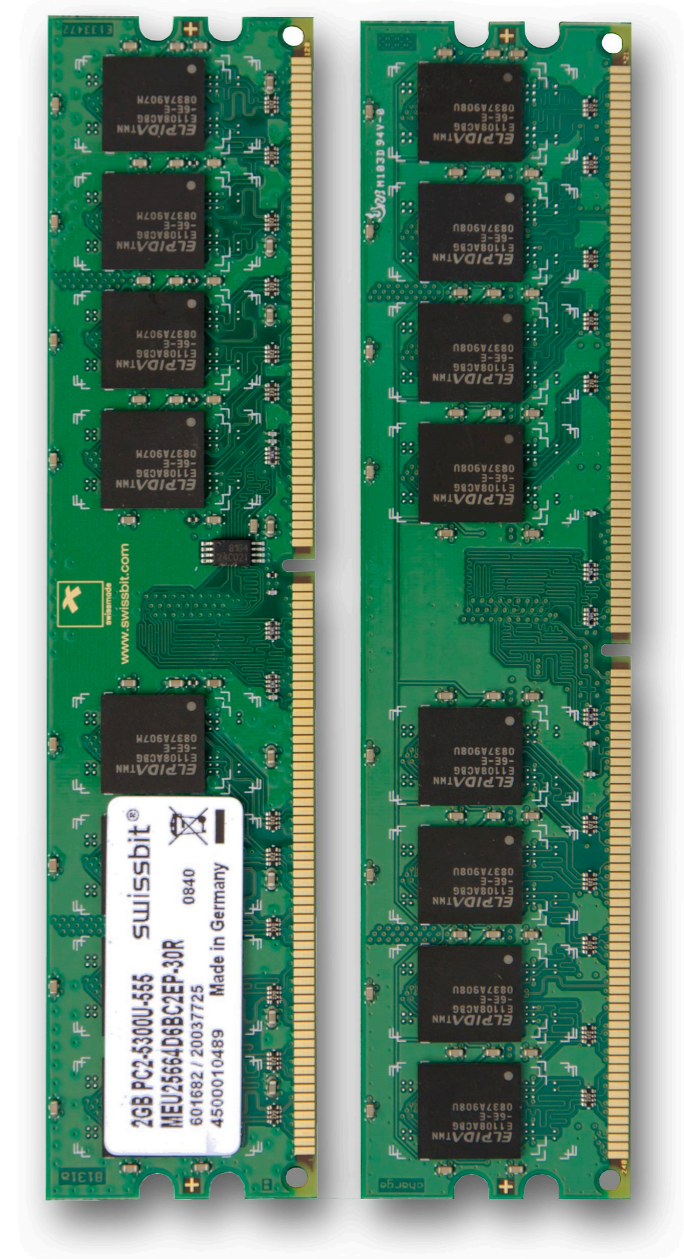
Out of PA space - Virtualisation

1. Does the user know whether the PFN came from TLB or memory?
 - No!
 - Does the user even know if paging is used?
2. Should the user know if the page came from memory or TLB or disk?
 - No!
 - The last time the process used the virtual address, it behaved like memory.
 - The next time the process uses the virtual address, it behaves like memory.

Out of PA space - Virtualisation

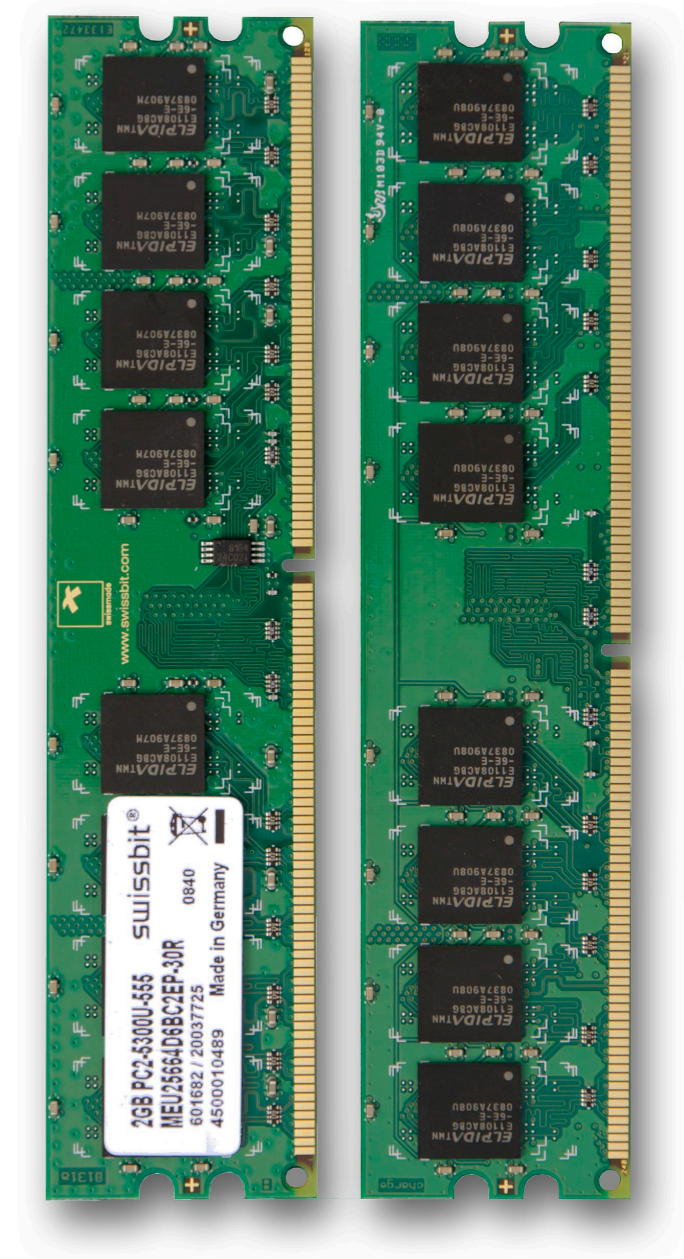
1. Does the user know whether the PFN came from TLB or memory?
 - No!
 - Does the user even know if paging is used?
2. Should the user know if the page came from memory or TLB or disk?
 - No!
 - The last time the process used the virtual address, it behaved like memory.
 - The next time the process uses the virtual address, it behaves like memory.
 - In between, whatever data was stored at that address must be preserved.

Swapping



Swapping

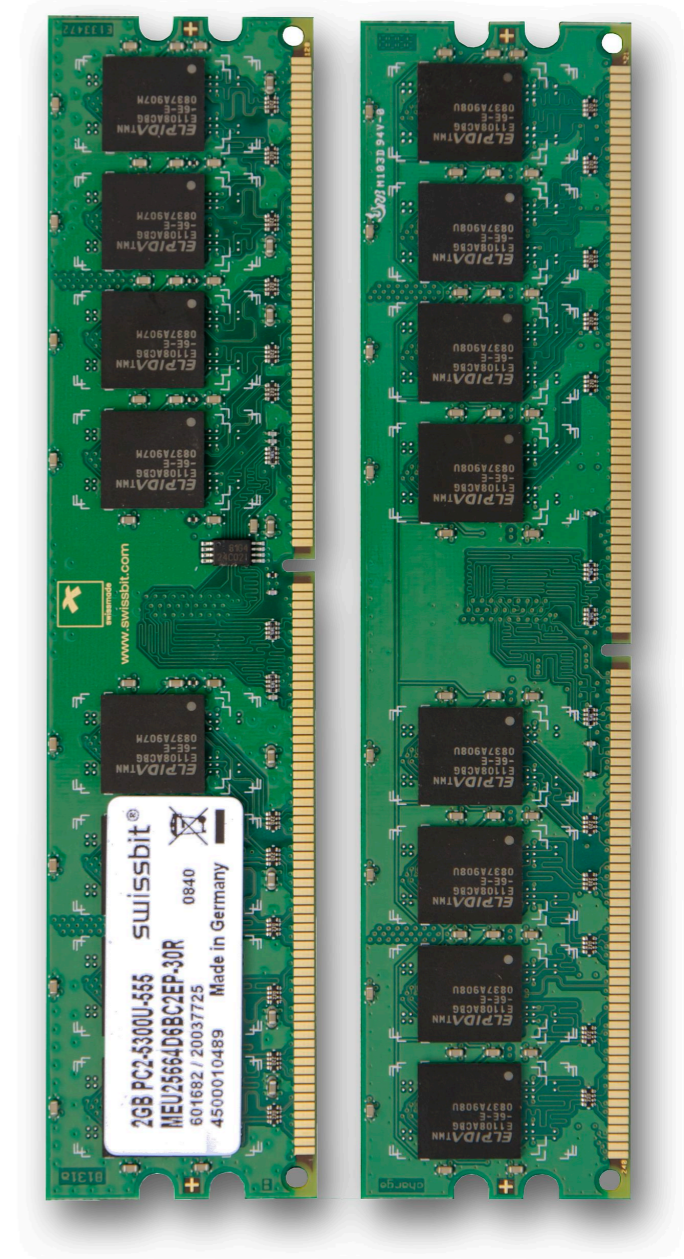
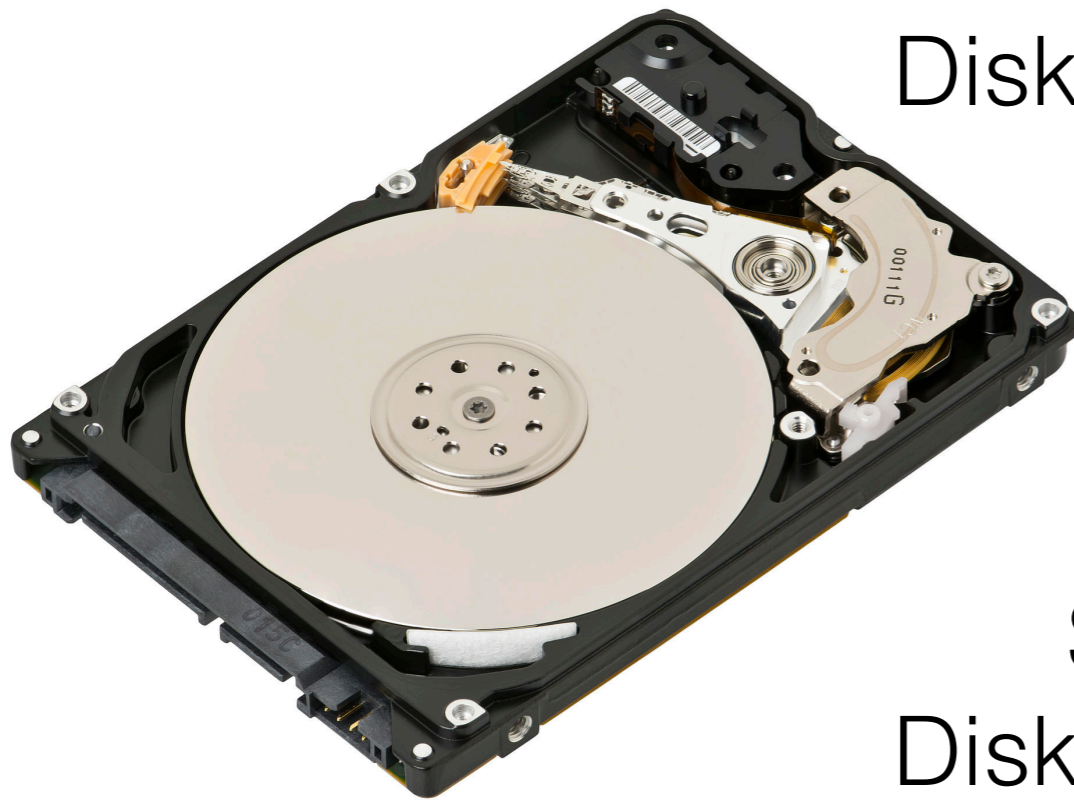
Swap in
Disk \longrightarrow Memory



Swapping

Swap in
Disk \longrightarrow Memory

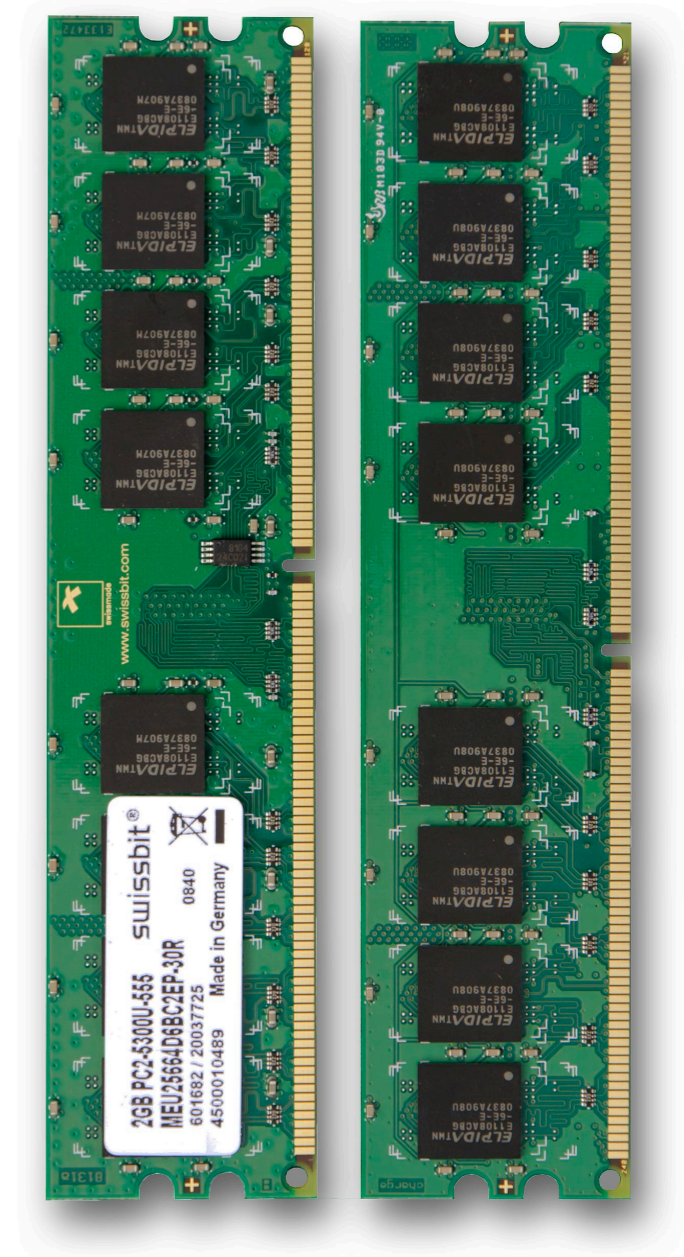
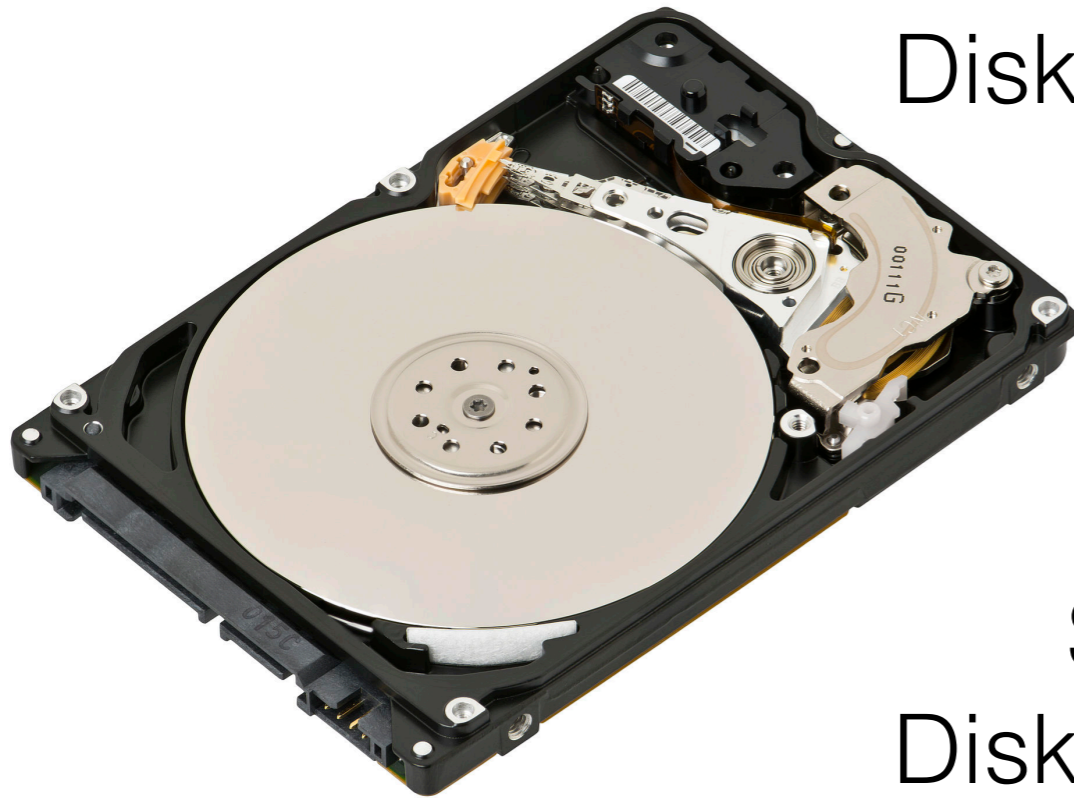
Swap out
Disk \longleftarrow Memory



Swapping

Swap in
Disk \longrightarrow Memory

Swap out
Disk \longleftarrow Memory

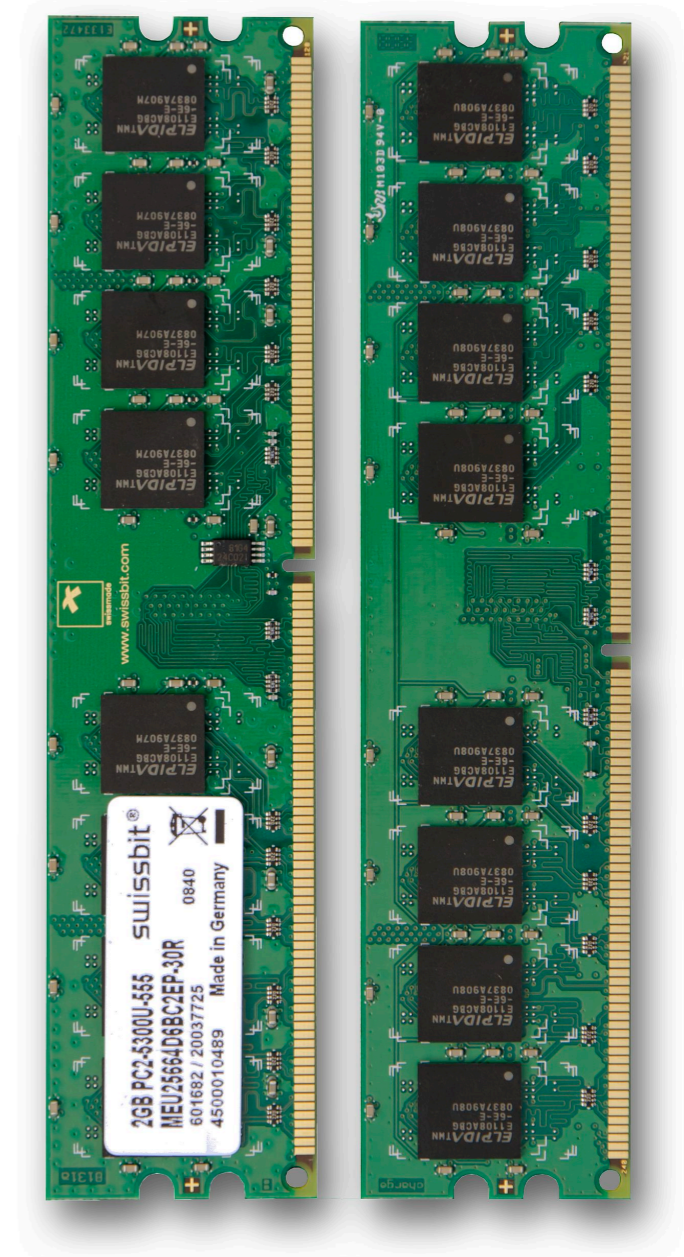
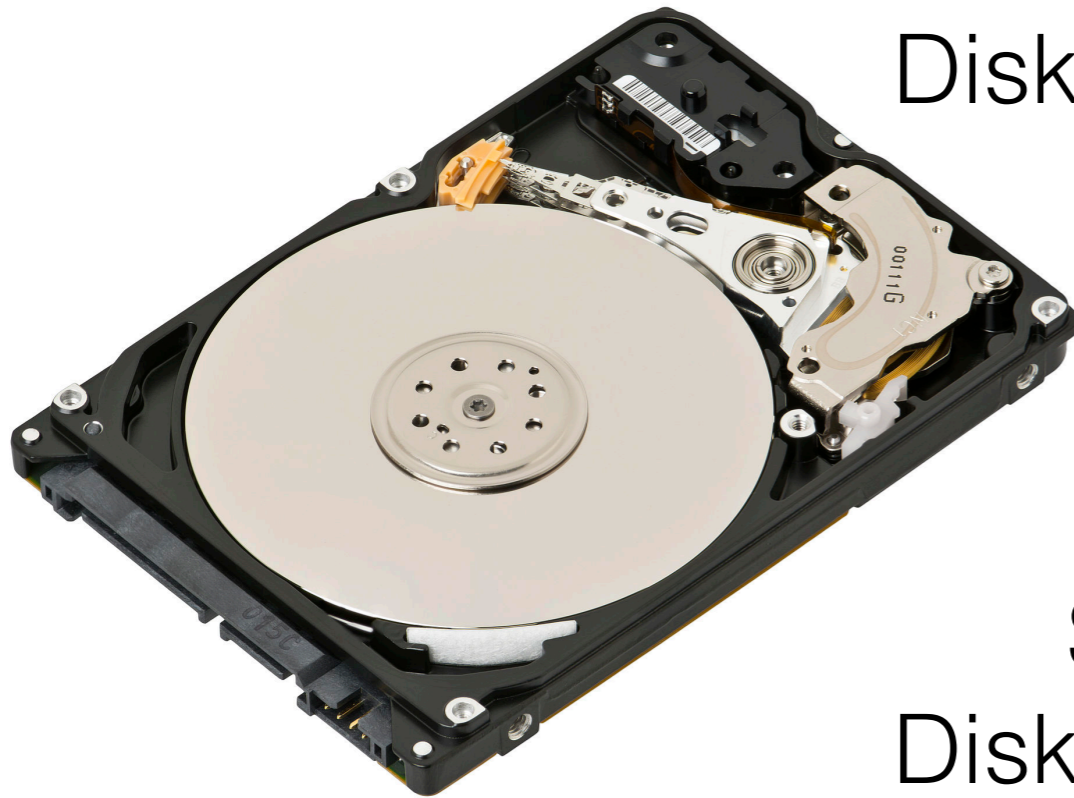


Done well : Memory as large as disk, as fast as RAM

Swapping

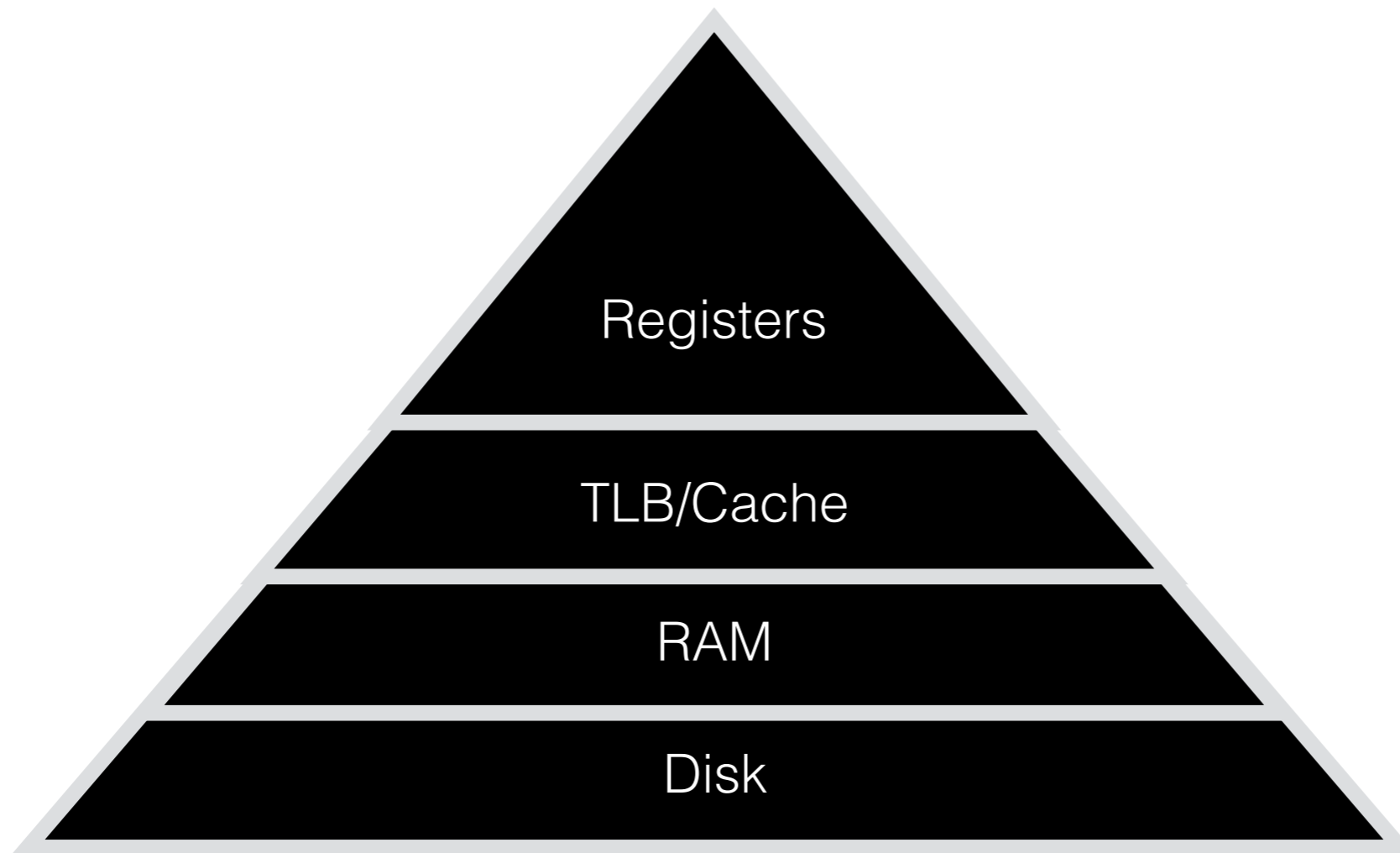
Swap in
Disk \longrightarrow Memory

Swap out
Disk \longleftarrow Memory

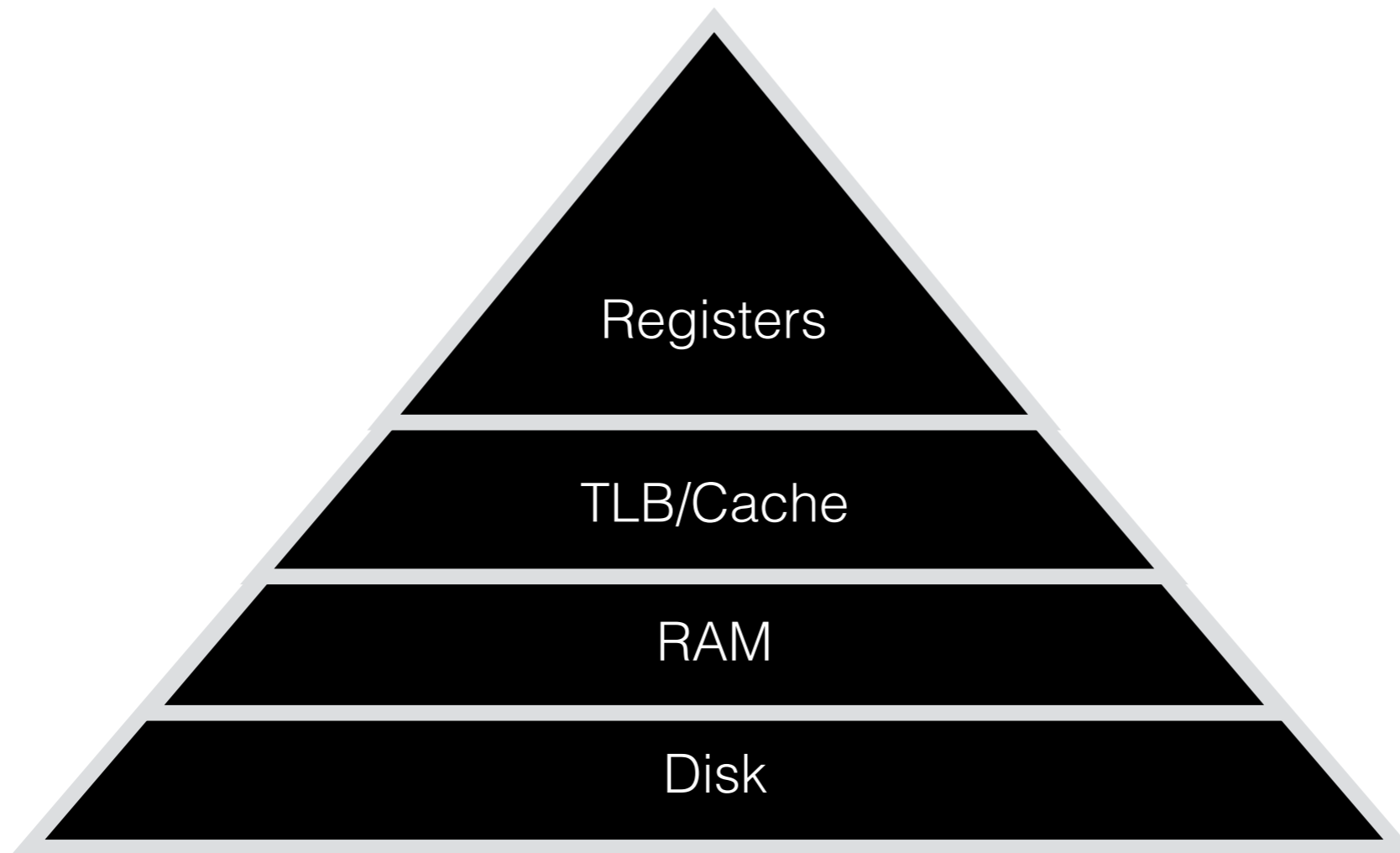


Done well : Memory as large as disk, as fast as RAM
Done bad : Memory as small as RAM, as slow as disk

TLB miss v/s Page Fault

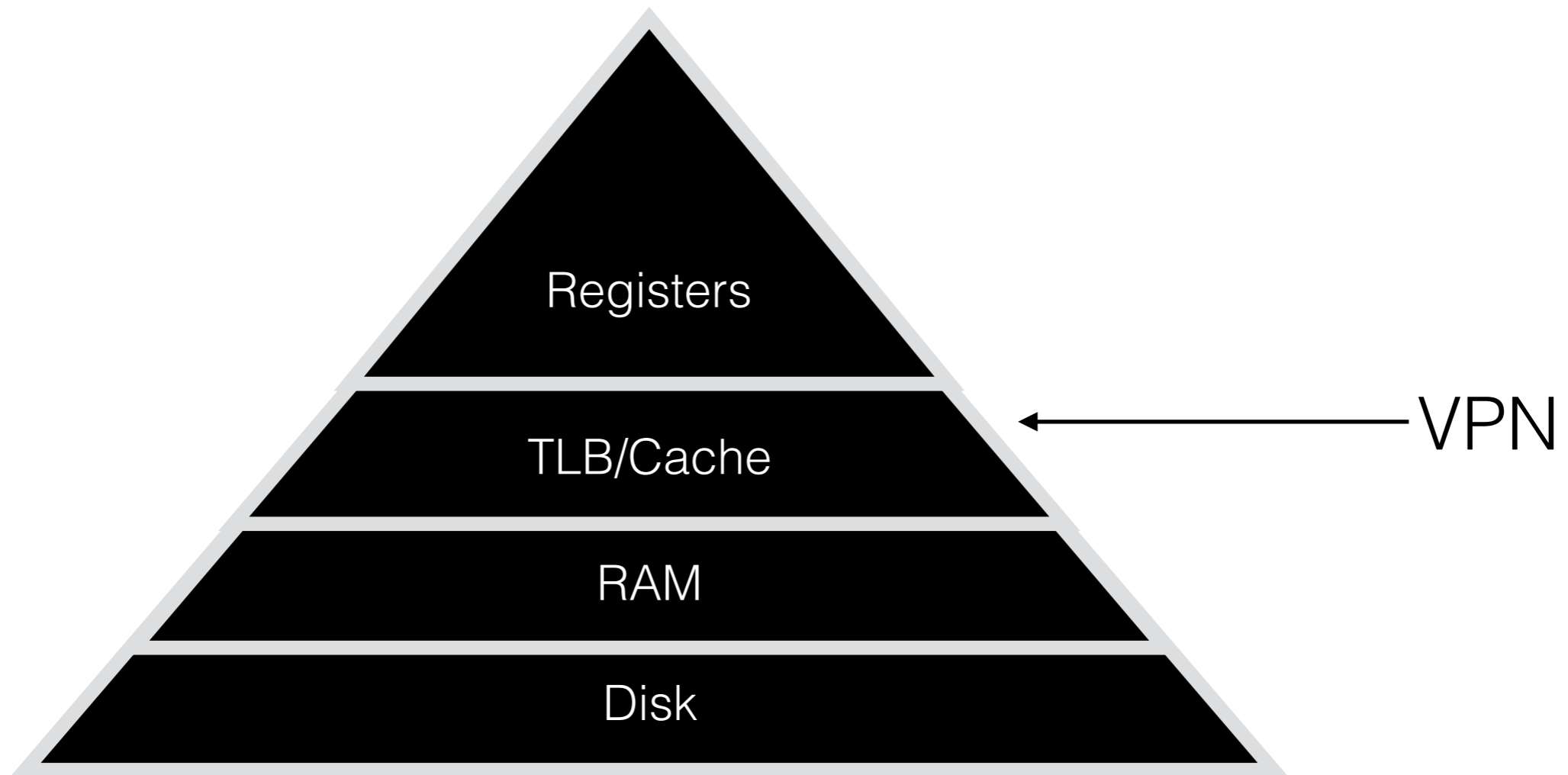


TLB miss v/s Page Fault

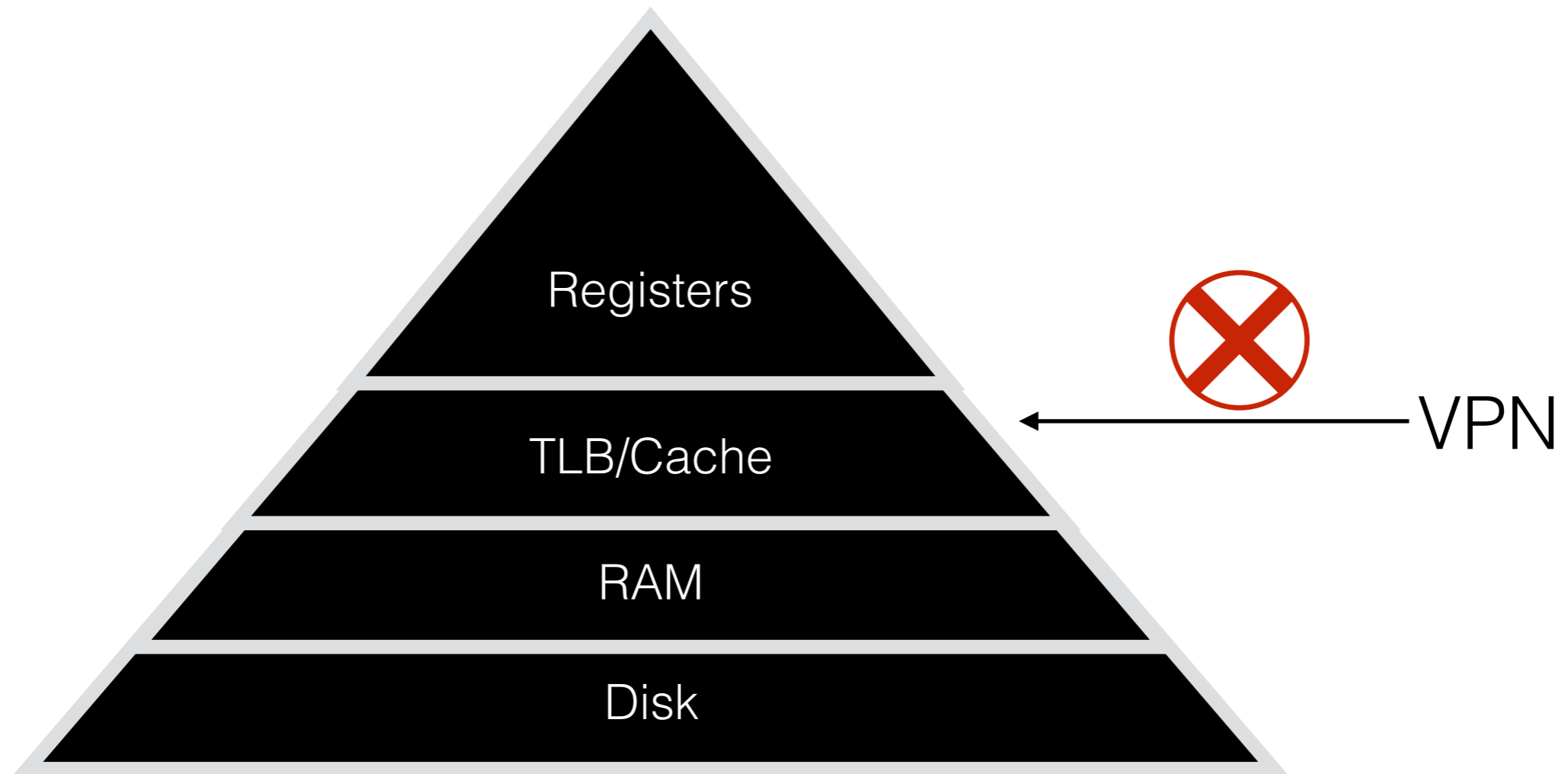


VPN

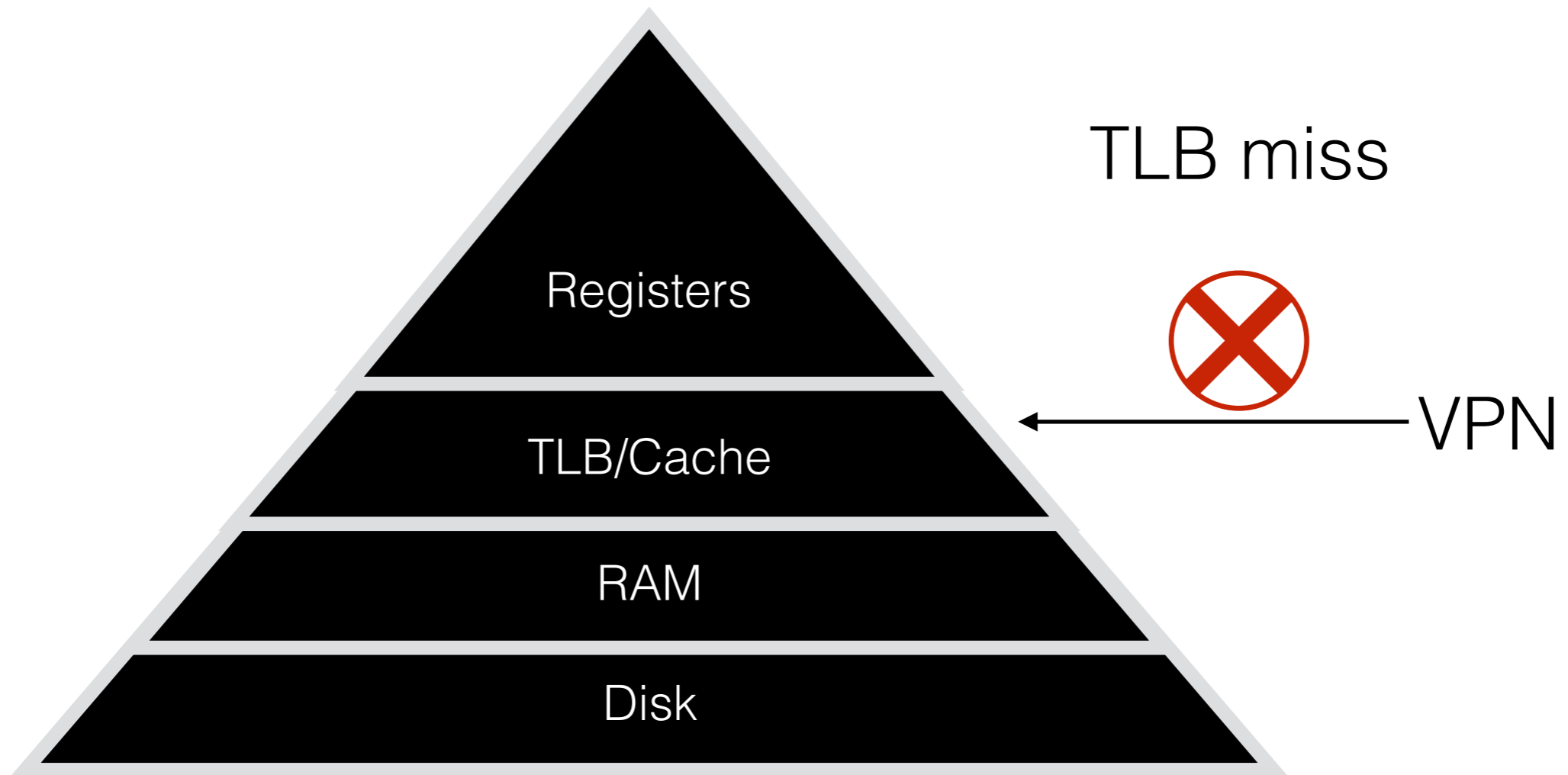
TLB miss v/s Page Fault



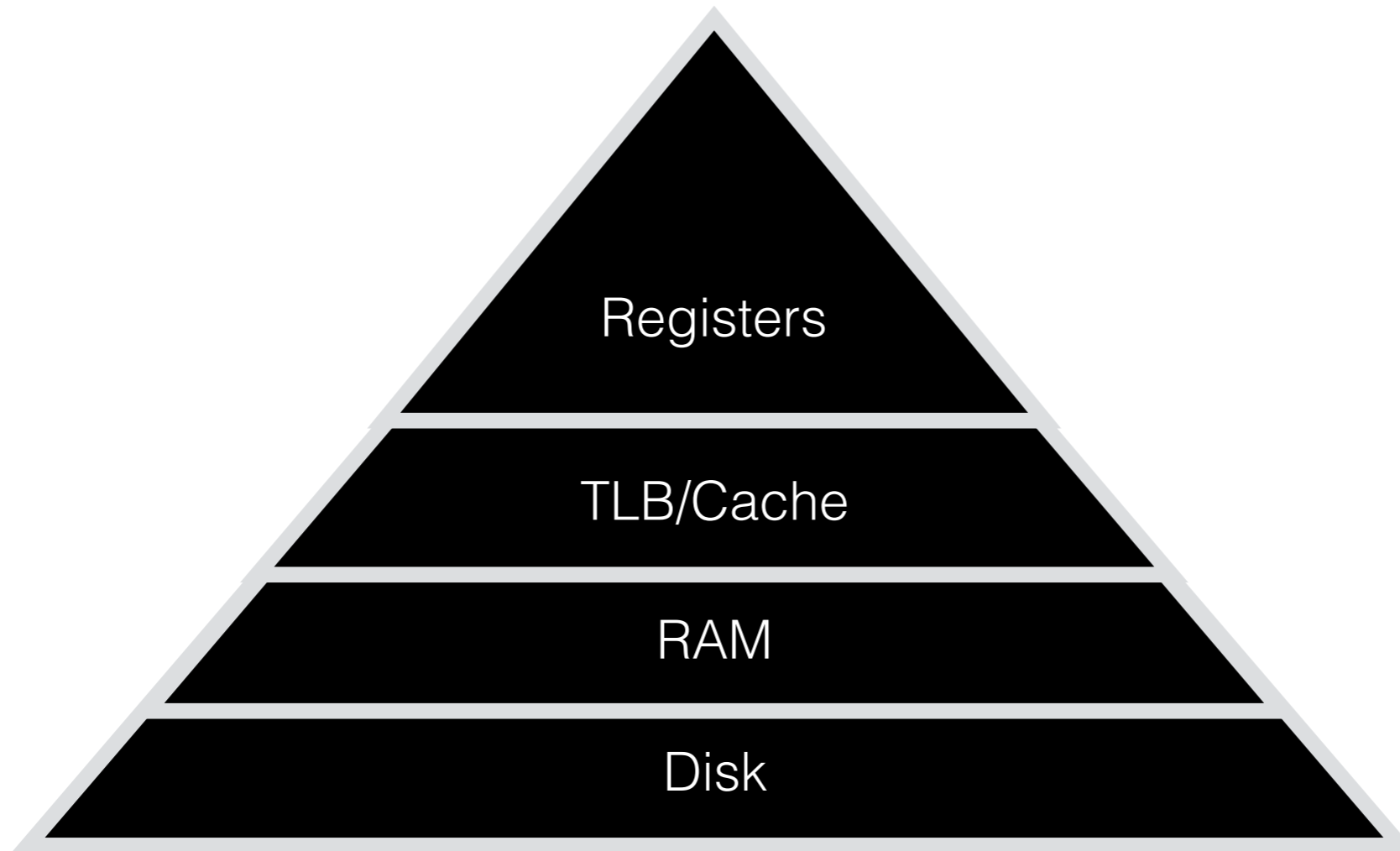
TLB miss v/s Page Fault



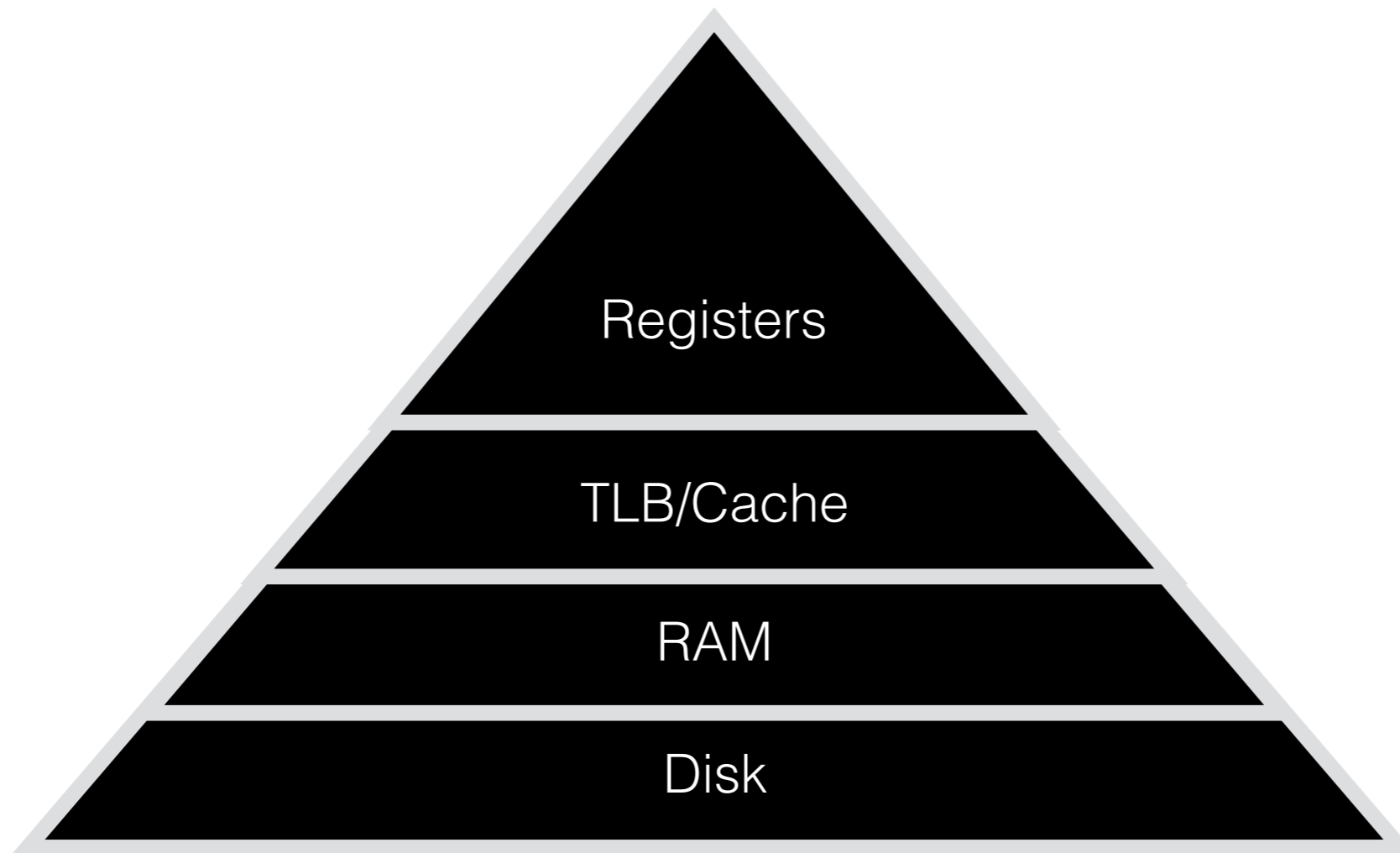
TLB miss v/s Page Fault



TLB miss v/s Page Fault

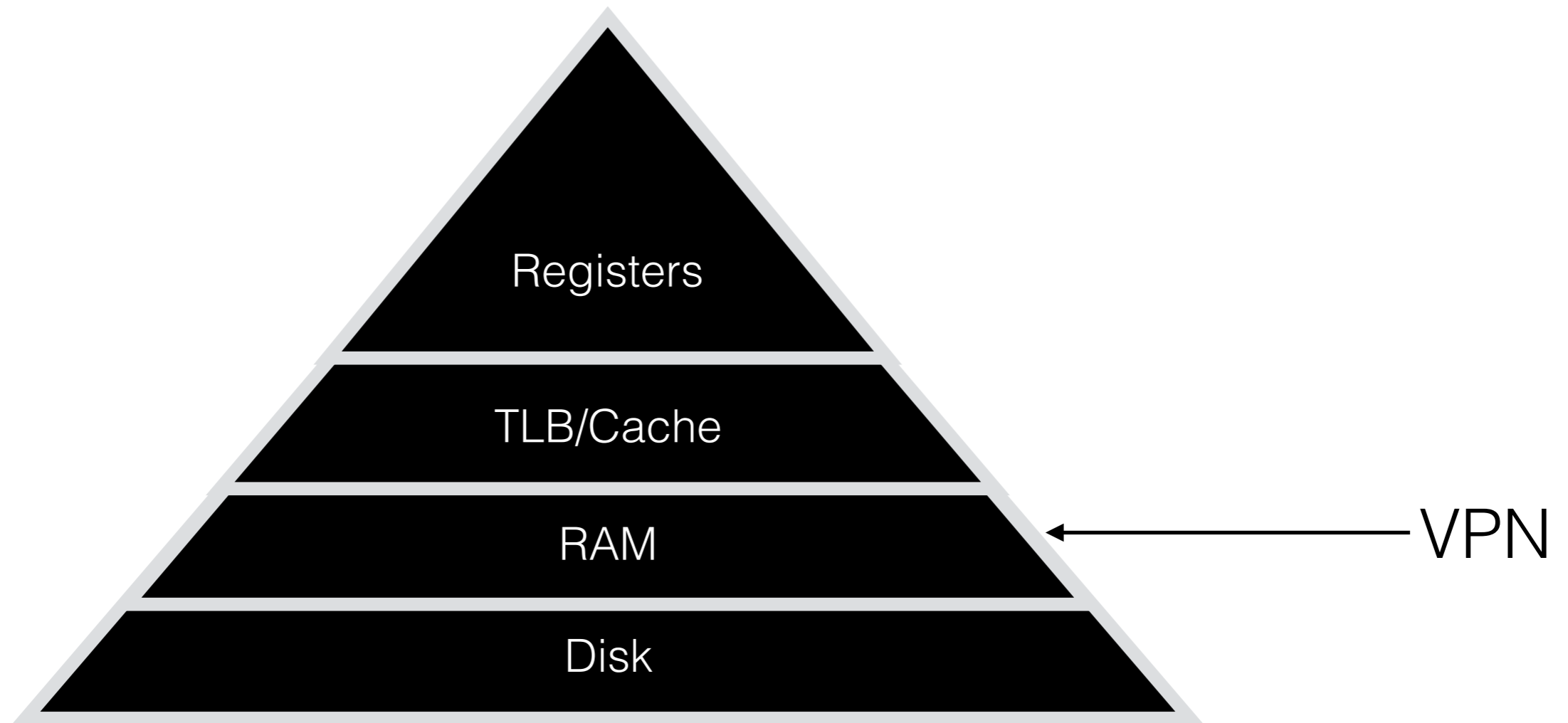


TLB miss v/s Page Fault

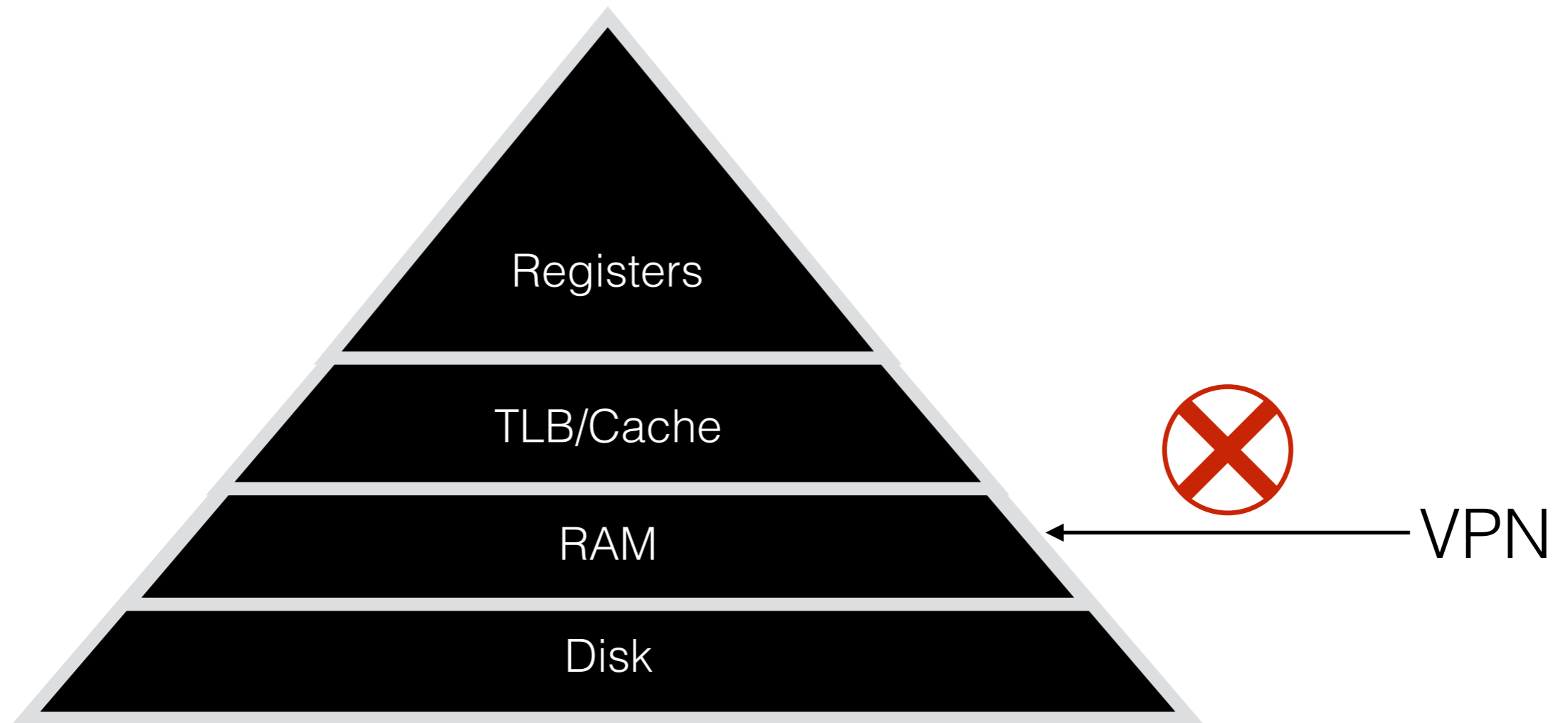


VPN

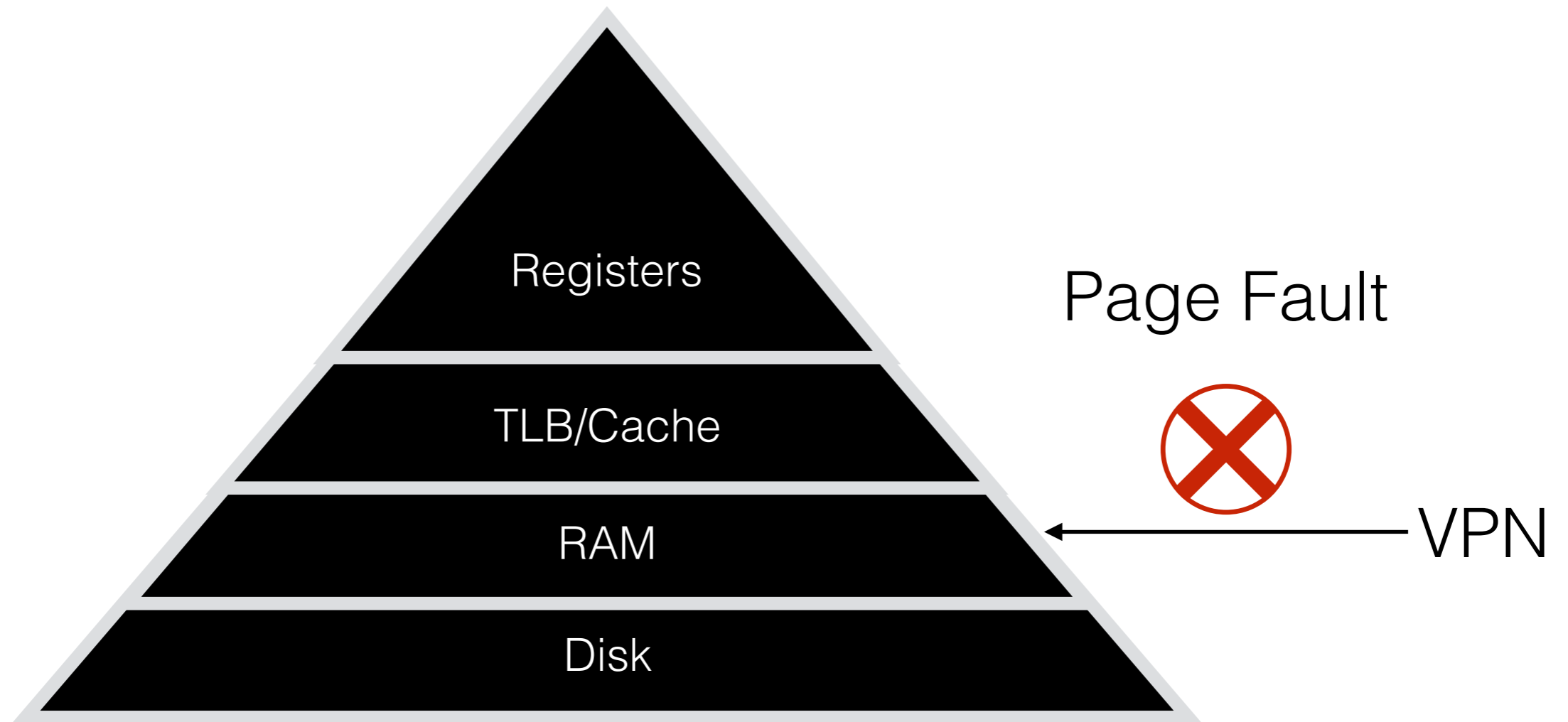
TLB miss v/s Page Fault



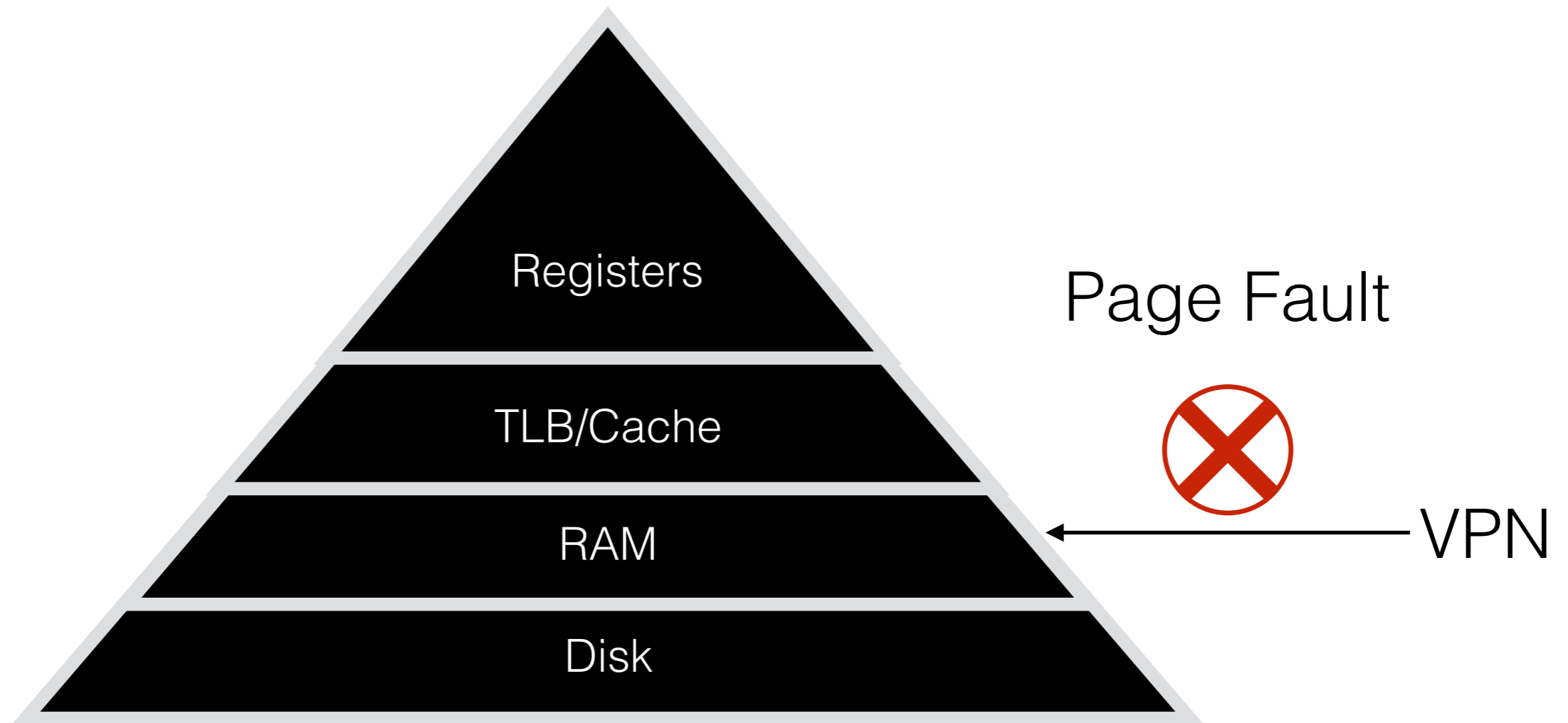
TLB miss v/s Page Fault



TLB miss v/s Page Fault

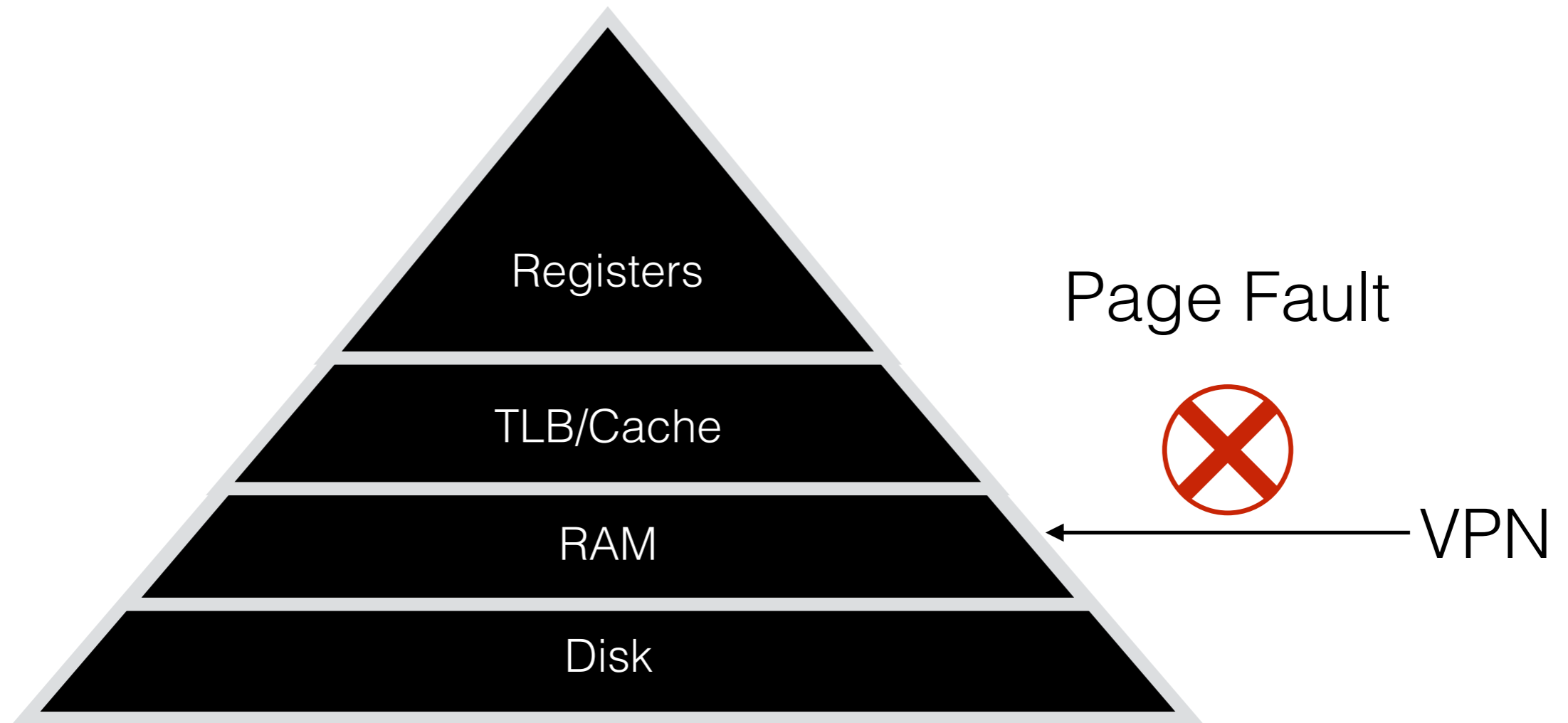


TLB miss v/s Page Fault



- Every page fault preceded by a TLB miss

TLB miss v/s Page Fault



- Every page fault preceded by a TLB miss
- Every TLB miss does not generate a page fault